21世纪软件工程专业规划教材

软件测试习题解析与实验指导

周元哲　编著



清华大学出版社

21世纪软件工程专业规划教材

软件测试习题解析与实验指导

周元哲　编著

清华大学出版社
北京

内 容 简 介

本书与《软件测试(第 2 版)》相配套,内容包括两部分。第 1 部分是习题解析,针对主教材的 8 章内容,给出每章的知识重点,精心设计了相应的习题,并给出了详细的解析和参考答案。第 2 部分是实验指导,主要包括黑盒测试(等价类划分法、边界值分析法、因果图)、白盒测试(逻辑覆盖、路径分析)以及 JUnit、TestDirector、LoadRunner、FindBugs、Bugzilla、Appium 等软件工具的使用。附录包括实验报告格式、软件测试相关文档模板、软件测试考试与竞赛简介。本书全面、系统地涵盖了当前业界测试领域的理论和实践知识,反映当前最新的软件测试理论、标准、技术和工具。

本书适合作为高等院校相关专业软件测试课程的教材或教学参考书,也可供从事计算机应用开发的各类技术人员参考,或用作全国计算机软件测评师考试、软件技术资格与水平考试的培训资料。

图书在版编目(CIP)数据

软件测试习题解析与实验指导/周元哲编著. —北京:清华大学出版社,2017
(21 世纪软件工程专业规划教材)
ISBN 978-7-302-47587-3

Ⅰ. ①软…　Ⅱ. ①周…　Ⅲ. ①软件—测试—高等学校—教学参考资料　Ⅳ. ①TP311.5

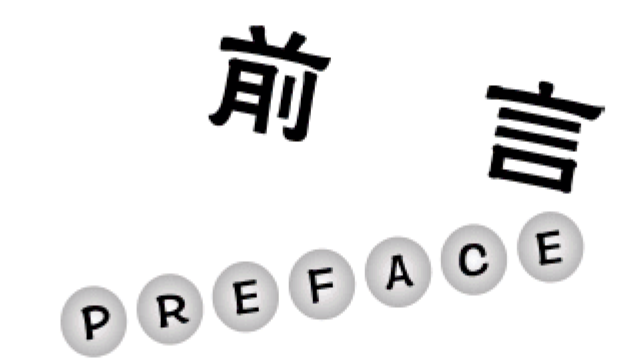
中国版本图书馆 CIP 数据核字(2017)第 154965 号

责任编辑:张　玥　战晓雷
封面设计:常雪影
责任校对:白　蕾
责任印制:沈　露

出版发行:清华大学出版社
网　　址:http://www.tup.com.cn, http://www.wqbook.com
地　　址:北京清华大学学研大厦 A 座　　**邮　　编**:100084
社 总 机:010-62770175　　**邮　　购**:010-62786544
投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈:010-62772015, zhiliang@tup.tsinghua.edu.cn
课件下载:http://www.tup.com.cn,010-62795954
印 装 者:三河市少明印务有限公司
经　　销:全国新华书店
开　　本:185mm×260mm　　**印　　张**:16.25　　**字　　数**:370 千字
版　　次:2017 年 10 月第 1 版　　**印　　次**:2017 年 10 月第1 次印刷
印　　数:1～2000
定　　价:39.50 元

产品编号:074448-01

前　言



本书与《软件测试(第2版)》相配套,在整个编写过程中结合作者多年从事软件工程和软件测试的教学经验,注重基本理论和基本实践的教学。在学习本书之前,需要一些先行课程作为本书的支撑,如计算机导论、程序设计语言、离散数学、软件工程等。

本书的写作目的是让学生在实践中增强动手能力,巩固理论知识,并用理论知识指导实践。本书意在让学生掌握软件测试的基本原理、基本方法、基本技术、基本标准和规范,培养学生的合作意识和团结精神,提高学生软件测试的综合能力。

本书内容包括软件测试习题解析、软件测试实验指导两部分。第1部分是习题解析,针对主教材8章,即软件测试概述、软件测试基本知识、黑盒测试、白盒测试、软件测试流程、性能测试、自动测试技术、软件测试管理,给出每章的知识重点,精心设计了相应的习题,并给出了详细的解析和参考答案。第2部分是软件测试实验指导,包括黑盒测试、白盒测试、单元测试软件JUnit、测试管理软件TestDirector、功能测试软件、性能测试软件LoadRunner、代码分析工具FindBugs、缺陷管理软件Bugzilla、移动测试软件Appium共9个实验,对每个实验都从实验目的及实验环境、实验内容、方案设计、测试数据及运行结果、源代码等方面进行介绍。附录包括实验报告格式、软件测试相关文档模板、软件测试考试与竞赛简介。为便于读者学习,在清华大学出版社网站(http://www.tup.com.cn)本书页面中提供了全国大学生软件测试大赛中使用的大角虫软件安装包。

本书由周元哲主编,其中,西北工业大学郑炜编写了第2部分的实验9。西安邮电大学计算机学院的王曙燕、邓万宇、孟伟君、舒新峰、张昕对本书的编写给予了大力的支持并提出了指导性意见,南京大学陈振宇、上海睿亚训软件技术服务公司王磊、韩伟以及清华大学出版社张玥编辑对本教材的写作大纲、写作风格等提出了很多宝贵的意见。本书在写作过程中参阅了大量中外文专著、教材、论文、报告及网络资料,在此向各位作者表示敬意和衷心的感谢。

本书内容精练,文字简洁,结构合理,综合性强,明确定位于面向初、中级读者,由入门起步,侧重提高,特别适合作为高等院校相关专业软件测试课程的教材或教学参考书,也可供从事计算机应用开发的各类技术人员参考,或用作全国计算机软件测评师考试、软件技术资格与水平考试的培训资料。

由于作者水平有限,时间紧迫,本书难免有不足之处,诚恳期待读者的批评指正,以使本书日臻完善。我们的电子信箱是 zhouyuanzhe@163.com。

作　者

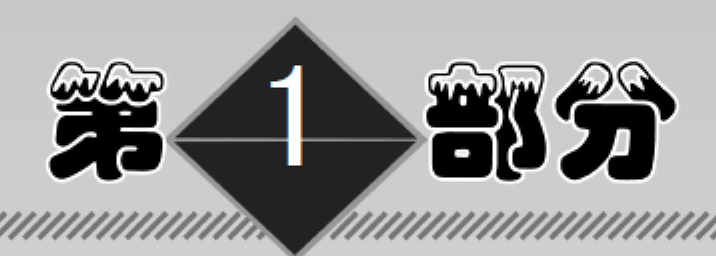
2017年6月

目 录

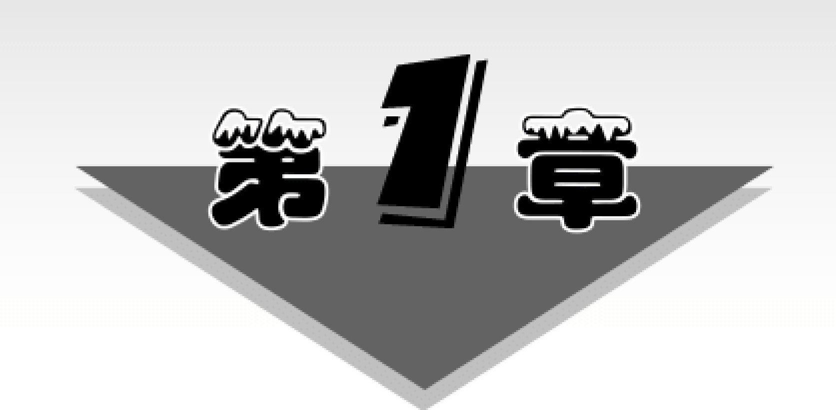
CONTENTS

第1部分 习题解析

第2部分　实验指导



习 题 解 析



软件测试概述

1.1 本章要求

- 了解软件发展史。
- 了解软件过程。
- 了解测试发展历程。
- 了解软件缺陷。
- 了解软件行业现状和测试职业。

1.2 本章知识重点

1. 软件开发与软件测试的关系

瀑布模型认为，测试是指在代码完成后、处于运行维护阶段之前，通过运行程序来发现程序代码或软件系统中的错误。因此，如果存在需求和设计上的缺陷，就会造成大量返工，增加软件开发的成本等。为了更早地发现问题，应该将测试延伸到需求评审、即设计审查活动中，即认为软件生命周期的每一阶段中都应包含测试。软件开发与软件测试的关系如图 1.1 所示。

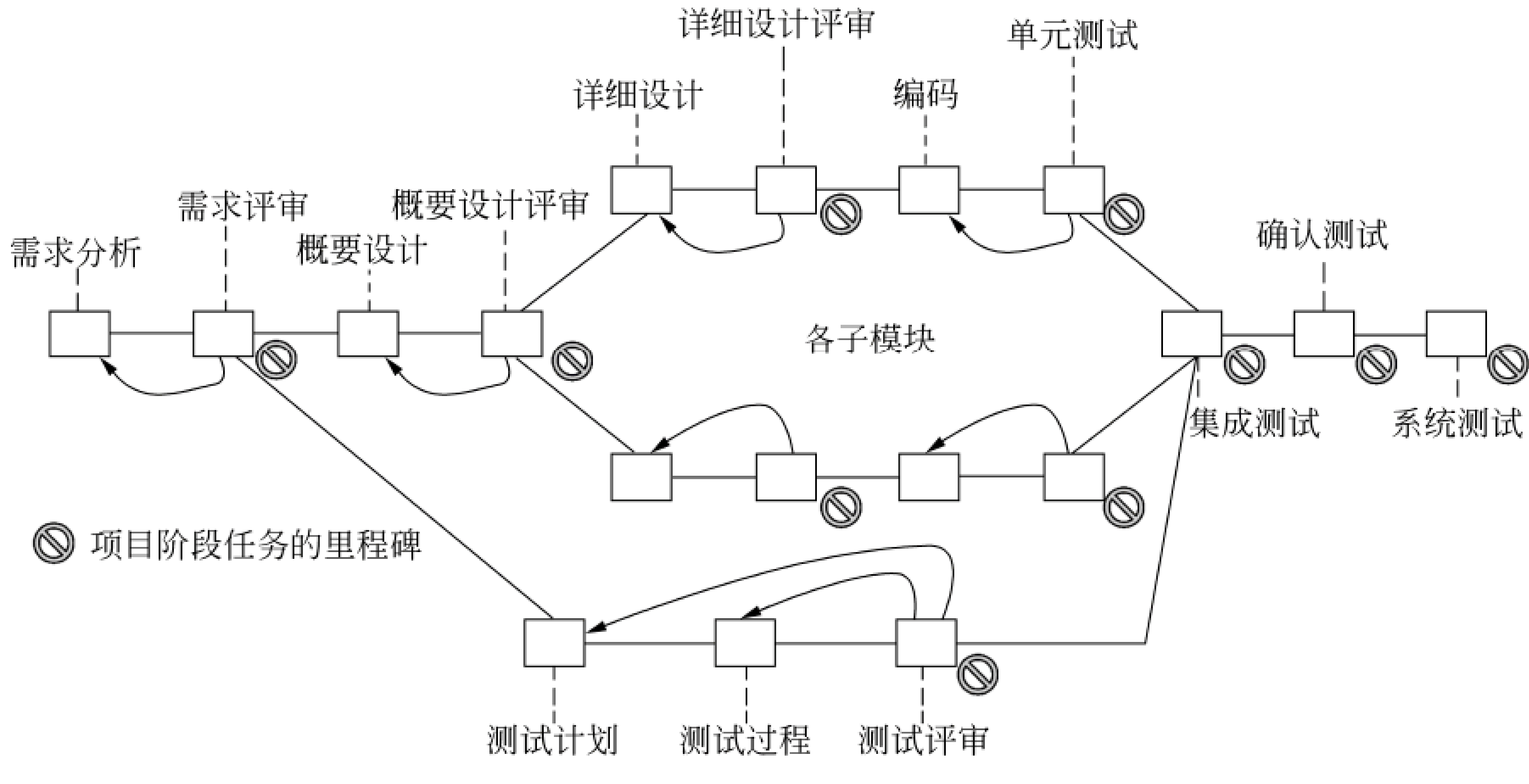


图 1.1　软件开发与软件测试的关系

2. 软件测试发展历程

软件测试伴随着软件的产生而产生。20 世纪 50 年代，软件测试等同于“调试”，目的是纠正软件的故障，常常由软件开发人员自己进行，主要是针对机器语言和汇编语言设计特定的测试用例，运行被测试程序，将所得结果与预期结果进行比较，从而判断程序的正确性。直到 1957 年，软件测试首次作为发现软件缺陷的活动与调试区分开来。1972 年，北卡罗来纳大学举行首届软件测试会议，确定软件测试是软件的一种研究方向。20 世纪 80 年代早期，一些软件测试的基础理论和实用技术开始形成，软件测试性质和内容也随之发生变化，测试不再是一个单纯的发现错误的过程，而是具有软件质量评价的内容。20 世纪 90 年代，测试能力成熟度模型(TCMM)、测试支持度模型(TSM)、测试成熟度模型(TMM)等一系软件测试相关理论提出。许多测试模型(如 V 模型等)产生，在单元测试、自动化测试等方面涌现了大量的软件测试工具。

3. 软件测试基本概念

定义 1：1983 年 IEEE(国际电子电气工程师协会)提出的软件工程标准术语中对软件测试的定义是“使用人工或自动手段来运行或测定某个系统的过程，其目的在于检验它是否满足规定的需求或弄清预期结果与实际结果之间的差别”。

定义 2：软件测试是根据软件开发各阶段的规格说明和程序的内部结构精心设计一批测试用例，并利用这些测试用例执行程序，以发现软件故障的过程。该定义强调测试的目的是寻找故障。

定义 3：软件测试是一种软件质量保证活动，其动机是通过一些经济有效的方法发现软件中存在的缺陷，从而保证软件质量。

4. 软件缺陷

软件缺陷一般有如下体现：

(1) 产品说明书中规定要做的事情，而软件没有实现。

例如，产品说明书要求计算器要实现加减乘除功能，但做出来的计算器不能进行除运算，这就是一个缺陷。

(2) 产品说明书中规定不要做的事情，而软件却实现了。

例如，产品说明书要求计算器除加减乘除功能外，其他的功能不用实现，但做出来的计算器不仅能进行加减乘除运算，还能进行乘方或三角函数运算，这也是一个缺陷。

(3) 产品说明书没有提到的事情，而软件却实现了。

例如，产品说明书要求计算器要实现加减乘除功能，但做出来的计算器还能进行乘方运算，这也是一个缺陷。

(4) 产品说明书中没有提到但是必须要做的事情，软件却没有实现。

例如，产品说明书要求计算器要实现加减乘除功能，但是没有提到在电量很低的情况下也能正常使用，而做出来的计算器在电量很低的时候计算错误，这也是一个缺陷。

(5) 软件很难理解，很难使用，速度极慢，测试人员站在最终用户的角度看到的问题

是平常的,但不是正确的。

例如,产品说明书要求计算器要实现加减乘除功能,但是按键使用的文字或标识不清楚,如“加”按键用“和”表示,或者计算 1+1 需要 1 分钟或者更长时间,这也是一个缺陷。

5. 软件行业现状

据资料显示,微软公司研发 Windows 2000 使用了 1700 名软件开发工程师、3200 名软件测试工程师。当前软件行业比较发达的国家与地区,如欧美、印度、以色列等,软件测试已经发展成为一个独立的产业,软件测试行业的产值几乎占软件行业总产值的 1/4,软件测试工程师和开发工程师的比例基本维持在 1∶1 左右,即 1 个软件开发工程师便需要辅以 1 个软件测试工程师。

我国软件测试也在不断发展,与国外软件测试主要存在如下差距:

(1) 测试的理解认识。国内软件企业普遍存在重开发、轻测试的情况,许多中小型软件企业没有软件测试部门及测试岗位。

(2) 测试过程的管理。国内软件企业普遍存在测试随意化、简单化的情况,未建立有效、规范的测试管理体系。

(3) 测试工具的使用。国内软件企业的测试普遍很少使用自动化测试工具。

(4) 测试人员的培养。我国软件测试领域目前十分缺乏各种层次的测试人才。

1.3 典型习题解析

1.3.1 选择题

1. 软件本身的特点和目前的软件开发模式使隐藏在软件内部的质量缺陷不可能完全避免。在下列关于导致软件质量缺陷的原因的描述中,不正确的是(　　)。

A. 软件需求模糊以及需求的变更,从根本上影响着软件产品的质量

B. 目前广为采用的手工开发方式难以避免出现差错

C. 程序员编码水平低下是导致软件缺陷最主要的原因

D. 软件测试技术具有缺陷

【答案】 A

2. (　　)是导致软件缺陷的最大原因。

A. 需求规格说明书　　B. 设计方案

C. 编写代码　　D. 测试计划

【答案】 A

3. 软件缺陷产生的原因是(　　)。

A. 交流不充分及沟通不畅,软件需求的变更,软件开发工具的缺陷

B. 软件的复杂性,软件项目的时间压力

C. 程序开发人员的错误,软件项目文档的缺乏

D. 以上都是

【答案】 D

4. 下列关于缺陷产生原因的叙述中，不属于技术问题的是（　　）。

A. 文档错误，内容不正确或拼写错误

B. 系统结构不合理

C. 语法错误

D. 接口传递不匹配，导致模块集成出现问题

【答案】 A

5. 导致软件缺陷的原因很多，①～④是可能的原因，其中最主要的是（　　）。

① 软件需求说明书不全面，不完整，不准确，而且经常更改

② 软件设计说明书不够详细

③ 软件操作人员的水平不足

④ 开发人员不能很好地理解需求说明书和沟通不足

A. ①②③　　B. ①③　　C. ②③　　D. ①④

【答案】 D

6. 下面有关软件缺陷的说法中错误的是（　　）。

A. 缺陷就是软件产品在开发中存在的问题

B. 缺陷就是软件维护过程中存在的错误、毛病等各种问题

C. 缺陷就是导致系统程序崩溃的错误

D. 缺陷就是系统所需实现的某种功能的失效和违背

【答案】 C

7. 功能或特性没有实现，主要功能部分丧失，次要功能完全丧失，这属于软件缺陷级别中的（　　）。

A. 致命缺陷　　B. 严重缺陷　　C. 一般缺陷　　D. 微小缺陷

【答案】 D

8. 实施缺陷跟踪的目的是解决以下（　　）问题。

A. 软件质量无法控制　　B. 问题无法量化

C. 重复问题接连产生

【答案】 ABC

9. 提高测试的有效性十分重要，“高产”的测试是指（　　）。

A. 用适量的测试用例运行程序，证明被测程序正确无误

B. 用适量的测试用例运行程序，证明被测程序符合相应的要求

C. 用少量的测试用例运行程序，发现被测程序尽可能多的错误

D. 用少量的测试用例运行程序，纠正被测程序尽可能多的错误

【答案】 C

10. 与设计测试数据无关的文档是（　　）。

A. 测试计划　　B. 需求说明书

C. 详细设计说明书　　D. 项目开发计划

【答案】 D

11. 测试人员的基本素质为(　　)。

A. 计算机专业技能　　B. 测试专业技能

C. 行业知识　　D. A、B、C

【答案】 D

12. 下列关于软件测试的叙述有错误的是(　　)。

A. 软件测试可以作为度量软件与用户需求间差距的手段

B. 软件测试的主要工作内容包括发现软件中存在的错误并解决存在的问题

C. 软件测试的根本目的是尽可能多地发现软件中存在的问题

D. 没有发现错误的测试也是有价值的

【答案】 D

13. 以下(　　)不属于软件缺陷。

A. 软件没有实现产品规格说明所要求的功能

B. 软件中出现了产品规格说明中规定不应该出现的功能

C. 软件实现了产品规格说明没有提到的功能

D. 软件实现了产品规格说明所要求的功能,但因受性能限制而未考虑可移植性问题

【答案】 D

14. 坚持在软件开发的各个阶段实施下列(　　)措施,才能在开发工程中尽早发现和预防错误。

A. 技术评审　　B. 程序测试　　C. 文档审查　　D. 管理评审

【答案】 A

15. 为了提高测试的效率,正确的做法是(　　)。

A. 选择发现错误可能性大的数据作为测试用例

B. 在完成程序的编码之后再指定软件的测试计划

C. 随机选取测试用例

D. 取一切可能的输入数据作为测试用例

【答案】 A

16. 软件生存周期过程中,修改错误代价最大的阶段是(　　)。

A. 需求阶段　　B. 设计阶段　　C. 编程阶段　　D. 发布运行阶段

【答案】 D

17. 下列能表达程序未按照预期运行但不会导致整体失效的是(　　)。

A. 故障　　B. 异常　　C. 缺点　　D. 失效

【答案】 B

18. 即使对程序的所有路径都进行了测试,程序也可能存在未能检查出来的缺陷,其原因可能是(　　)。

A. 程序可能会因为缺少某些路径而存在问题

B. 即使是穷举路径测试也决不能保证程序符合其设计规格说明

C. 穷举路径测试也可能不会暴露数据敏感错误

D. A、B、C

【答案】 D

19. 以下选项中不属于软件缺陷状态的是(　　)。

A. 激活状态　　B. 非激活状态　　C. 一致状态　　D. 已修正状态

【答案】 C

20. 程序中存在的某种破坏正常运行能力的问题、错误或者隐藏的功能缺陷属于(　　)。

A. 缺陷　　B. 故障　　C. 失效　　D. 缺点

【答案】 A

21. 问题还没有解决,测试人员又报告了新缺陷,或验证后缺陷仍然存在,这些缺陷所处的状态是(　　)。

A. 激活状态　　B. 非激活状态　　C. 已修正状态　　D. 关闭状态

【答案】 A

22. 下列不属于软件本身的原因产生的缺陷的是(　　)

A. 算法错误　　B. 语法错误

C. 文档错误　　D. 系统结构不合理

【答案】 C

23. 证实在一个给定的外部环境中软件的逻辑正确性是(　　)。

A. 验证　　B. 确认　　C. 测试　　D. 调试

【答案】 B

24. 对于一个软件的各种需求,要确定其关键性类型,定义关键性级别的依据是(　　)。

A. 系统任务　　B. 安全性　　C. 技术复杂性　　D. A、B、C

【答案】 D

25. 下列不属于动态分析的软件行为是(　　)。

A. 屏幕仿真　　B. 分支执行分析　　C. 结构分析　　D. 建模

【答案】 C

26. 不用执行程序,目的是收集有关程序代码的结构信息,这一过程是(　　)。

A. 性能测试　　B. 静态分析　　C. 增量测试　　D. 大突击测试

【答案】 B

27. 编码阶段的测试目标是确定程序代码的质量,代码质量的确定依据是(　　)。

A. 设计规格说明可跟踪到程序相应的代码,程序代码可跟踪到设计需求

B. 分析程序接口并与接口文档相对照

C. 执行程序评估工作,分析程序是否是设计说明的正确翻译,是否与程序编码标准相符

D. A、B、C

【答案】 D

28. 下列情况表明出错处理功能有错误和缺陷的是(　　)。

A. 显示的错误与实际遇到的错误不符　B. 显示的错误信息难以理解

C. 对异常处理的不得当　D. A、B、C

【答案】 D

29. 下列可以作为软件测试对象的是(　　)。

A. 需求规格说明书　B. 软件设计规格说明

C. 源程序　D. A、B、C

【答案】 D

30. 软件测试是采用(　　)执行软件的活动。

A. 测试用例　B. 输入数据　C. 测试环境　D. 输入条件

【答案】 A

31. 导致软件缺陷的最大原因是(　　)。

A. 软件需求说明书　B. 设计方案

C. 编码　D. 维护

【答案】 A

32. 在下列描述中,关于一个软件缺陷状态完整变化的错误描述是(　　)。

A. 打开—修复—关闭　B. 打开—关闭

C. 打开—保留　D. 激活—修复—重新打开

【答案】 D

1.3.2 判断题

1. 通过软件测试,可以发现软件中所有潜伏的错误。(×)

2. 软件测试是对软件规格说明、软件设计和编码的最全面也是最后的审查。(√)

3. 如果测试过程中没有发现任何错误,则说明软件没有错误。(×)

4. 软件测试小组人数越多越好。(×)

5. 测试是为了验证该软件正确地实现了用户的需求。(×)

6. 程序效率的提高主要应通过选择高效的算法来实现。(√)

7. 尽量用公共过程或子程序代替重复的代码段。(×)

8. 在程序运行之前无法评估其质量。(×)

9. 缺乏有力的方法学指导和有效的开发工具的支持,往往是产生软件危机的原因之一。(√)

10. 目前的绝大多数软件都不适合采用快速原型技术。(×)

11. 下列哪些活动是项目?

(1) 探索火星生命迹象。(√)

(2) 向部门经理进行月工作汇报。(×)

(3) 开发新版本的操作系统。(√)

(4) 每天的卫生保洁。(×)

(5) 组织超级女声决赛。(√)

(6) 一次集体婚礼。 (√)
12. 发现错误多的程序模块,残留在模块中的错误也多。 (√)
13. 项目立项前测试人员不需要提交任何工件。 (×)
14. 程序中隐藏错误的概率与其已发现的错误数成正比。 (√)
15. 代码评审是检查源代码是否达到模块设计的要求。 (√)

1.3.3 简答题

1. 什么是软件?

【答】 软件是一系列按照特定顺序组织的计算机数据和指令的集合。一般来讲,软件划分为编程语言、系统软件、应用软件和介于这两者之间的中间件。其中系统软件为计算机使用提供最基本的功能,但是并不针对特定应用领域。而应用软件则恰好相反,根据用户和所服务的领域提供不同的功能。

2. 软件经过了哪几个发展阶段?

【答】 20 世纪 50 年代初期至 60 年代中期为软件发展的第一阶段,又称为程序设计阶段。此时硬件已经通用化,而软件的生产却是个体化。软件产品为专用软件,规模较小,功能单一,开发者即使用者,软件只有程序,无文档。软件设计在人们的头脑中完成,形成了"软件等于程序"的错误观念。

第二阶段从 20 世纪 60 年代中期至 70 年代末期,称为程序系统阶段。此时多道程序设计技术、多用户系统、人机交互式技术、实时系统和第一代数据库管理系统出现,出现了专门从事软件开发的"软件作坊",软件广泛应用,但软件技术和管理水平相对落后,导致"软件危机"出现。

第三阶段称为软件工程阶段,1968 年,北大西洋公约组织的计算机科学家在联邦德国召开国际会议,讨论软件危机问题,正式提出并使用"软件工程"概念,标志软件工程诞生。

第四阶段是从 20 世纪 80 年代中期至今,出现了客户端/服务器体系结构,特别是 Web 技术和网络分布式对象技术、面向服务架构等用于解决传统对象模型中无法解决的异构和耦合问题。

3. 什么是容错软件?

【答】 容错软件的定义有如下 4 种:

(1) 在一定程度上对自身错误的作用具有屏蔽能力的软件。
(2) 在一定程度上能从错误状态自动恢复到正常状态的软件。
(3) 在发生错误时,仍然能在一定程度上完成预期的功能的软件。
(4) 在一定程度上具有容错能力的软件。

4. 软件错误、软件缺陷、软件故障、软件失效各是什么?

【答】

(1) 软件错误是在软件生存期内不可接受的人为错误,一般是指编写代码时出现的

错误,称为 bug。

(2) 软件缺陷是软件错误的结果,是软件产品预期属性的偏离现象,其范围较广,涵盖了软件错误、不一致性问题、功能需求定义缺陷和产品设计缺陷等。缺陷分为过错缺陷和遗漏缺陷。如果把某些信息输入到不正确的表示中,就是过错缺陷;如果没有输入正确信息,就是遗漏缺陷。

(3) 软件失效是指软件运行时由于软件缺陷产生的不可接受的外部行为结果。

(4) 软件故障是软件失效所表现出来的现象,是指软件运行过程中出现的不可接受的内部状态。

5. 软件测试和程序测试有什么异同?

【答】 软件测试是对软件计划、软件设计、软件编码进行查错和纠错的活动(包括代码执行活动与人工活动)。测试的目的是找出软件设计开发全周期中各个阶段的错误,以便分析错误的性质与位置而加以纠正,纠正过程可能涉及改正或重新设计相关的文档活动。

程序测试是对编码阶段的语法错、语义错等进行查找的执行活动。纠正编码中的错误的执行活动称程序调试。通过查找编码错与纠正编码错来保证算法的正确实现。

软件测试及调试与程序测试及调试相同之处都是查错与纠错的活动。差别在查错与纠错的范围不同,软件测试覆盖软件生存周期整个阶段,而程序测试则仅限于编码阶段。

6. 从技术角度来看,软件测试发现错误的现象有哪些?

【答】

(1) 现象与原因所处的位置可能相距甚远。

(2) 当其他错误得到纠正时,这一错误所表现出的现象可能会暂时消失,但并未实际排除。

(3) 现象实际上是由一些非错误原因(例如舍入不精确)引起的。

(4) 现象可能是由于一些不容易发现的人为错误引起的。

(5) 错误是由于时间序列问题引起的,与处理过程无关。

(6) 现象是由于难于精确再现的输入状态(例如实时应用中输入顺序不确定)引起的。

(7) 现象可能是周期性出现的。在软、硬件结合的嵌入式系统中常常遇到。

7. 什么是软件测试?

【答】 软件测试是指通过一定方法或工具对被测试对象进行检验,以发现被测试对象具有某种属性或者存在某些问题的过程。

8. 软件测试发展经历了几个历程?

【答】 软件测试伴随着软件的产生而产生。早期软件开发过程中,软件规模小,复杂程度低,软件开发过程相当混乱无序,软件测试含义也比较窄,等同于“调试”,目的是纠正

软件的故障,常常由软件开发人员自己进行。

直到 1957 年,软件测试首次作为发现软件缺陷的活动与调试区分开来。1972 年,北卡罗来纳大学举行首届软件测试会议,John Good Enough 和 Susan Gerhart 在 IEEE 上发表《测试数据选择的原理》,确定软件测试是软件的一种研究方向。1979 年,Glenford Myers 在《软件测试艺术》一书中提出"测试是为发现错误而执行的一个程序或者系统的过程"。

20 世纪 80 年代早期,测试不再是一个单纯发现错误的过程,而是具有软件质量评价的内容。1983 年,Bill Hetzel 在《软件测试完全指南》中指出,测试是以评价一个程序或者系统属性为目标的任何一种活动,是对软件质量的度量。

20 世纪 90 年代,软件测试工具开始运用。1996 年,测试能力成熟度(TCMM)、测试支持度(TSM)、测试成熟度(TMM)等一系软件测试相关理论提出。到了 2002 年,Rick 和 Stefan 在《系统的软件测试》一书中对软件测试做了进一步描述:测试是为了度量和提高软件的质量,对软件进行工程设计、实施和维护的整个生命周期过程。

近 20 年来,随着计算机和软件技术飞速发展,软件测试技术的研究也取得了很大的突破。许多测试模型(如 V 模型等)产生,单元测试、自动化测试等方面涌现了大量的软件测试工具。

9. 国内软件测试行业与国外的差距表现在哪几方面?

【答】 国内软件测试与国外主要存在如下差距:

(1) 测试的理解认识。国内软件企业普遍存在重开发、轻测试的情况,将测试置于从属地位,没有认识到软件项目完成不仅取决于开发人员,更取决于测试人员。国内许多中小型软件企业没有软件测试部门,有些甚至不设置软件测试的岗位。

(2) 测试过程的管理。国内软件企业普遍存在测试随意化、简单化的情况,未建立规范的测试管理体系。

(3) 测试工具的使用。国内软件企业的测试普遍极少使用自动化测试工具。

(4) 测试人员的培养。软件测试领域目前十分缺乏人才,首先是测试人员角色定位不合理,其次是缺乏专家级测试人才。

10. 排错是什么?

【答】 排错是查找、分析和纠正错误的过程。测试是用人工或自动方法执行或评价软件系统的过程,以验证是否满足规定的需求,或识别期望结果和实际结果之间有无关联。测试的任务是尽可能多地发现软件的缺陷和错误,而排错的任务是进一步诊断和改正程序中潜伏的缺陷和错误。一般来说,排错有两类活动:一是确定程序中可疑缺陷或错误的确切性质和位置;二是对程序(设计、编码)进行修改,排除潜在的缺陷或错误。

11. 关于测试的认识有哪些误区?

【答】 关于软件测试有如下误区。

误区一:使用了测试工具,就是进行了有效的测试。

误区二:软件开发完成后进行软件测试。

误区三：软件发布后发现质量问题，是测试人员的问题。
误区四：软件测试要求不高，随便找个人就行。
误区五：项目进度吃紧时少做些测试，时间充裕时多做测试。
误区六：软件测试是低级工作，开发人员才是软件高手。

12. 缺陷管理的主要内容是什么？

【答】 软件缺陷就是软件产品中存在的问题，最终表现为用户所需要的功能没有完全实现，不能满足或不能全部满足用户的需求。

13. 缺陷报告编制的基本原则是什么？

【答】 编制缺陷报告的基本原则可以概括为5C：
Correct（准确）：每个组成部分的描述准确，不会引起误解。
Clear（清晰）：每个组成部分的描述清晰，易于理解。
Concise（简洁）：只包含必不可少的信息，不包括任何多余的内容。
Complete（完整）：包含复现该缺陷的完整步骤和其他本质信息。
Consistent（一致）：按照一致的格式书写全部缺陷报告。

14. 软件缺陷划分成哪几个等级？

【答】 软件的缺陷具有如下等级：
(1) 致命。数据被破坏，数据丢失，系统崩溃，系统无法运行。
(2) 严重。处理结果不正确，流程不对，性能不能满足要求。
(3) 一般。不会影响整个系统的运行性能。
(4) 微小。操作不方便，有错别字，界面布局不合理，难以理解等。
(5) 建议。界面、描述更改等。

15. 一个优秀的测试工程师需要具备的素质有哪些？

【答】 优秀的测试工程师需要具备以下素质：
- 探索精神。软件测试员不害怕进入陌生环境。
- 追求完美。他们力求完美，但是知道无法企及时，不去强求。
- 不懈努力。不停尝试，他们不会心存侥幸，而是尽一切可能去寻找。
- 判断准确。要觉得测试内容，测试时间以及看到的内容是否是真正的软件缺陷。
- 老练稳重。不害怕坏消息，知道怎样和不够老练的程序员合作。
- 有说服力。善于表达观点。



软件测试基本知识

2.1 本 章 要 求

- 了解软件测试目的和原则。
- 理解软件测试分类。
- 掌握软件测试模型。
- 掌握测试用例。
- 了解软件测试充分性与测试停止标准。

2.2 本章知识重点

1. 软件测试目的

一般认为,软件测试的目的往往包含如下内容:

(1) 测试并不仅仅是为了找出错误。通过分析错误产生的原因和错误的发生趋势,可以帮助项目管理者发现当前软件开发过程中的缺陷,以便及时改进。

(2) 测试能够帮助测试人员设计出有针对性的测试方法,改善测试的效率和有效性。

(3) 没有发现错误的测试也是有价值的,完整的测试是评定软件质量的一种方法。

测试的目标就是以最少的时间和人力找出软件中潜在的各种错误和缺陷,证明软件的功能和性能与需求说明相符。此外,实施测试收集到的测试结果数据为可靠性分析提供了依据。

Grenford J. Myers 关于软件测试提出以下观点:

(1) 测试是为了证明程序有错,而不是证明程序无错误。

(2) 一个好的测试用例在于它能发现至今未发现的错误。

(3) 一个成功的测试是发现了至今未发现的错误的测试。

2. 软件测试原则

(1) 应当把“尽早地和不断地进行软件测试”作为软件开发者的座右铭。

(2) 严防寄生虫现象

(3) 严防杀虫剂现象

(4) 并非所有的软件缺陷都能修复
(5) 难以说清的软件缺陷
(6) 需求规格说明书不断变化
(7) 合理的设计测试用例
(8) 软件测试充分性准则

3. 软件测试分类

软件测试分类如图 2.1 所示。

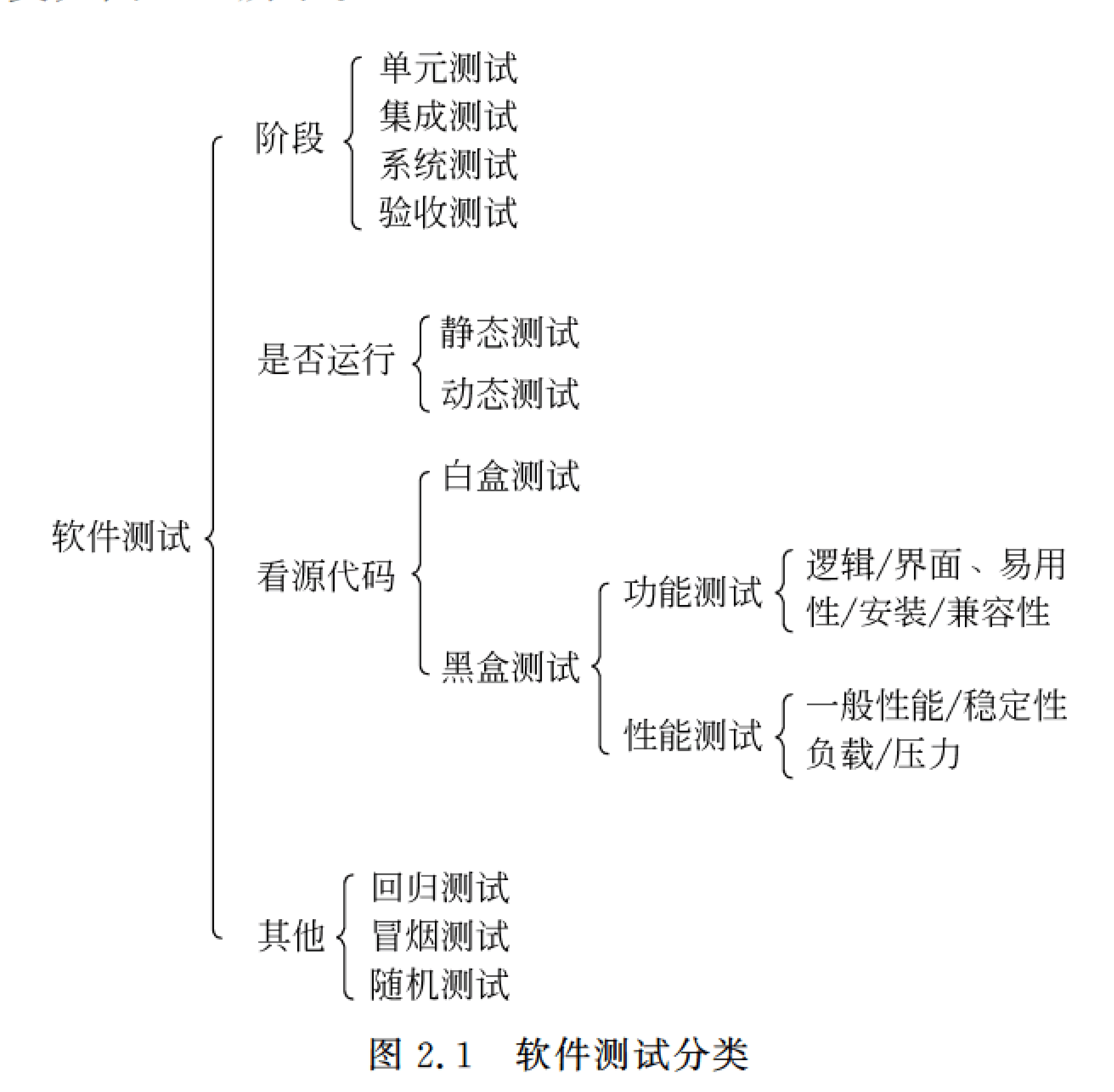


图 2.1 软件测试分类

4. 软件测试模型

软件测试模型用于指导软件测试的实践，通常有 V 模型、W 模型、H 模型等。

1) V 模型

V 模型作为最典型的测试模型，反映了测试活动与开发活动的关系，标明测试过程中存在的不同级别，并清楚描述测试的各个阶段和开发过程的各个阶段之间的对应关系。V 模型左侧是开发阶段，右侧是测试阶段。开发阶段先从定义软件需求开始，然后把需求转换为概要设计和详细设计，最后形成程序代码。测试阶段是在代码编写完成以后，先作单元测试，然后是集成测试、系统测试和验收测试。在 V 模型中，单元测试对应详细设计，也就是说，单元测试用例和详细设计文档一起实现；而集成测试对应概要设计，其测试用例是根据概要设计中模块功能及接口等实现方法编写的。依次类推，测试计划在软件需求完成后就开始进行，完成系统测试用例的设计等。V 模型如图 2.2 所示。

V 模型仅把测试过程作为在需求分析、概要设计、详细设计及编码之后的一个阶段，主要针对程序进行寻找错误的活动，而忽视了测试活动对需求分析、系统设计等活动的验

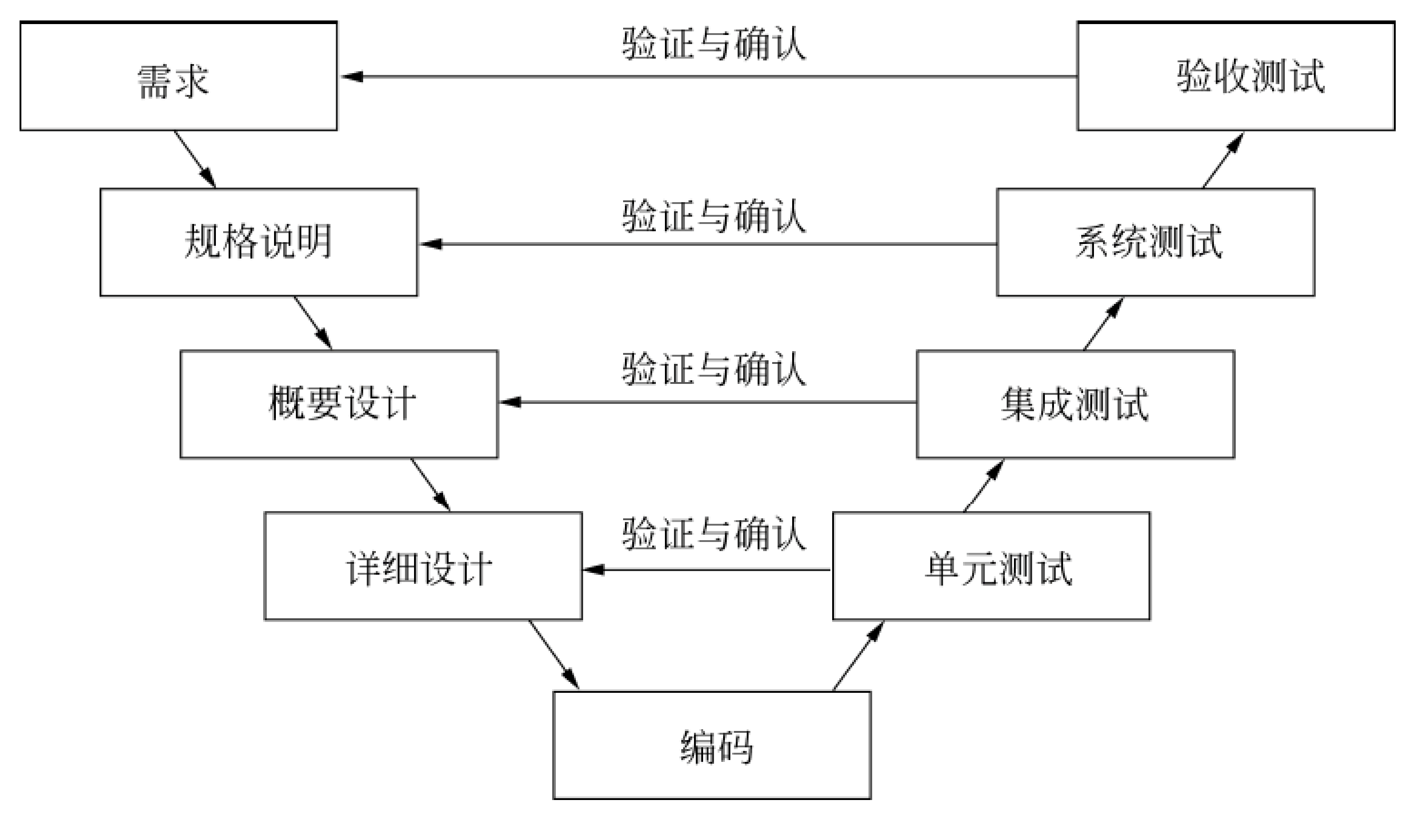


图 2.2 V 模型示意图

证和确认的功能。

2）W 模型

相对于 V 模型而言，W 模型增加了软件各开发阶段中应同步进行的验证和确认活动。如图 2.3 所示，W 模型由两个 V 字形的模型组成，分别代表测试与开发过程，明确表示出了测试与开发的并行关系。

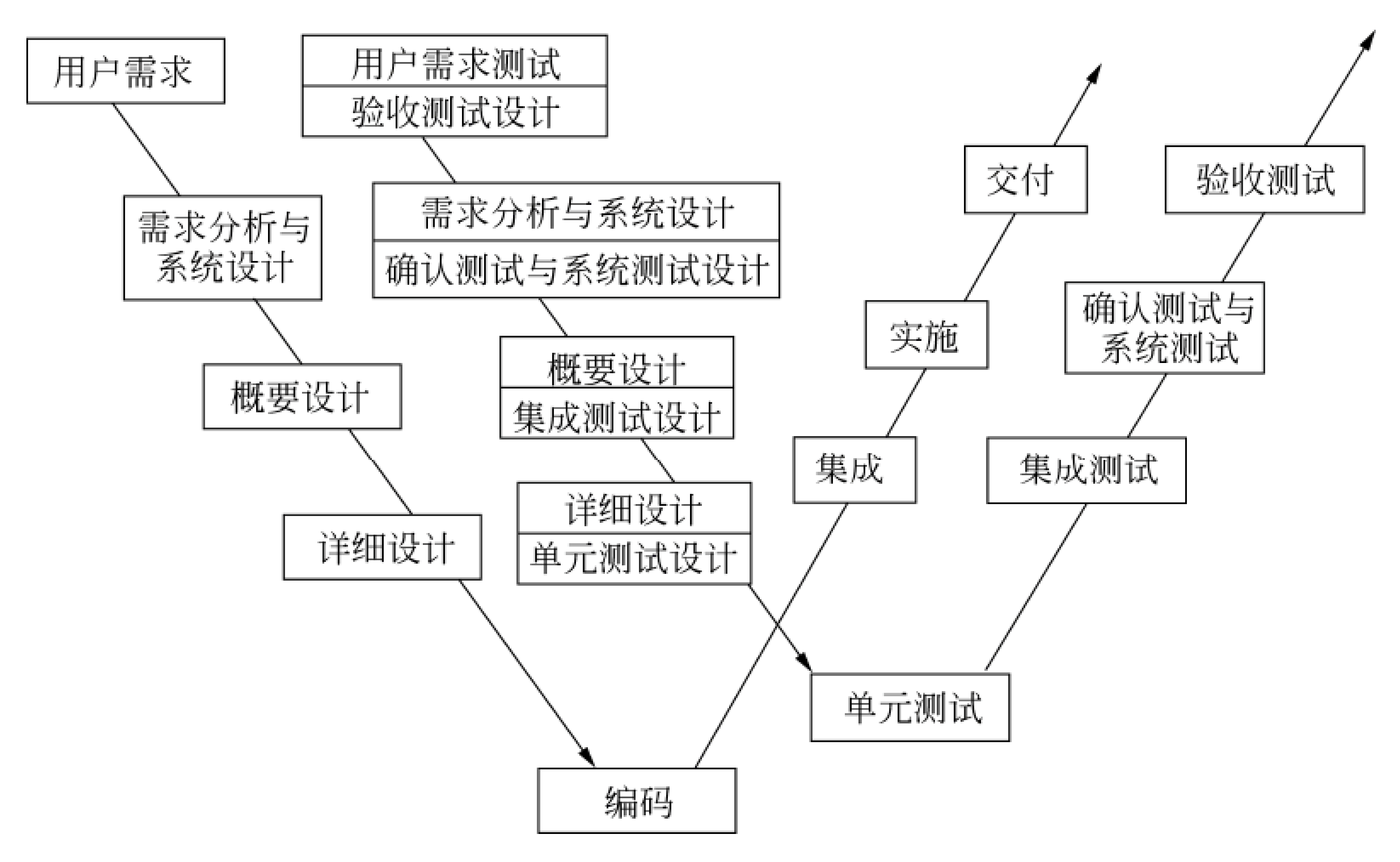


图 2.3 W 模型示意图

W 模型强调测试伴随着整个软件开发周期，测试的对象不仅仅是程序，需求、设计等同样要测试，也就是说，测试与开发同步进行。W 模型有利于尽早地发现问题，只要相应的开发活动完成，就可以开始测试。例如，需求分析完成后，测试就应该参与到对需求的验证和确认活动中，以尽早地找出缺陷所在。同时，对需求的测试也有利于及时了解项目难度和测试风险，及早制定应对措施，从而减少总体测试时间，加快项目进度。在 W 模型中，需求、设计、编码等活动被视为串行，测试和开发活动保持着一种线性的前后关系，上

一阶段结束，才开始下一个阶段工作，因此，W 模型无法支持迭代开发模型。

3）H 模型

V 模型和 W 模型都认为软件开发是需求、设计、编码等一系列串行的活动，而事实上，这些活动在大部分时间内可以交叉，因此，相应的测试也不存在严格的次序关系，单元测试、集成测试、系统测试之间会反复迭代。正因为 V 模型和 W 模型存在这样的问题，H 模型将测试活动完全独立出来，使得测试准备活动和测试执行活动清晰地体现出来。H 模型如图 2.4 所示。

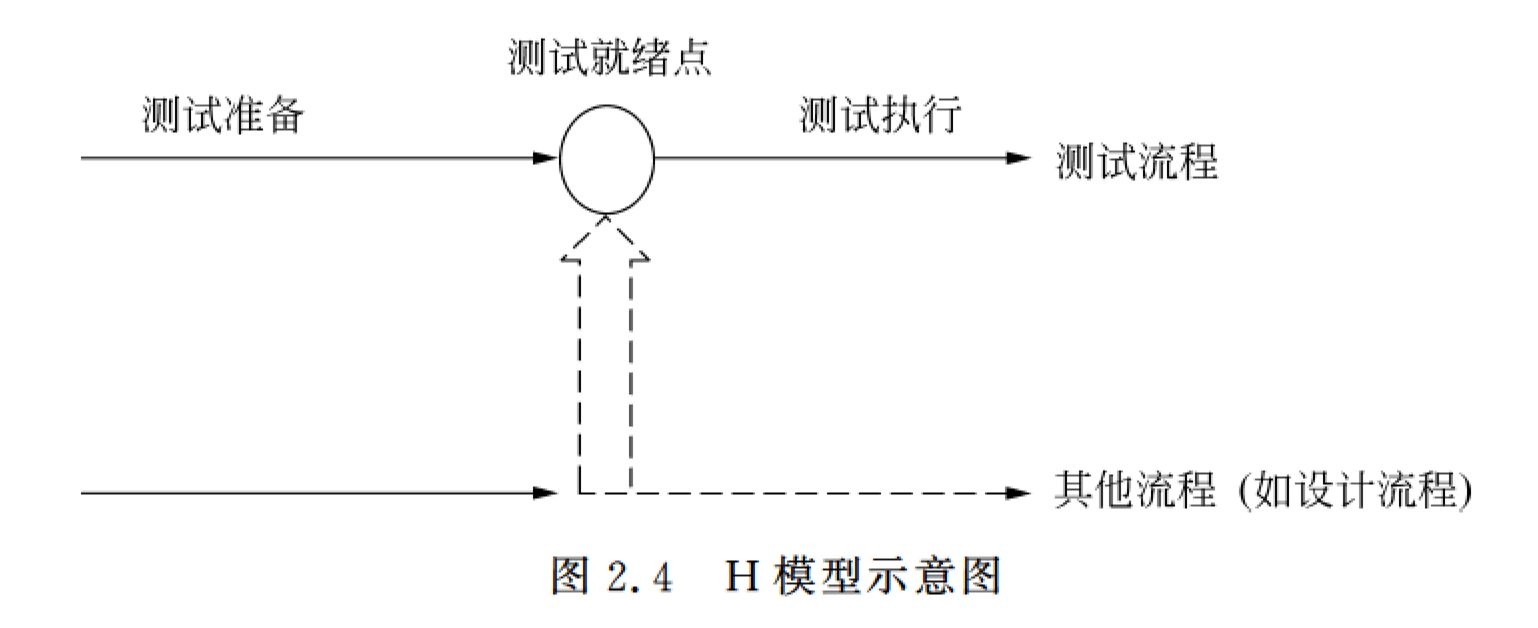


图 2.4　H 模型示意图

图 2.4 仅仅显示了整个测试生命周期中某个层次的“微循环”。H 模型揭示了软件测试作为一个独立的流程贯穿于软件整个生命周期，与其他流程并发地进行，并指出软件测试要尽早准备，尽早执行。不同的测试活动可以按照某个次序先后进行，也可能是反复的，只要某个测试达到准备就绪点，测试执行活动就可以开展。

4）X 模型

由于 V 模型未能体现出测试设计、测试回溯的过程，因此，出现了 X 测试模型，如图 2.5 所示。

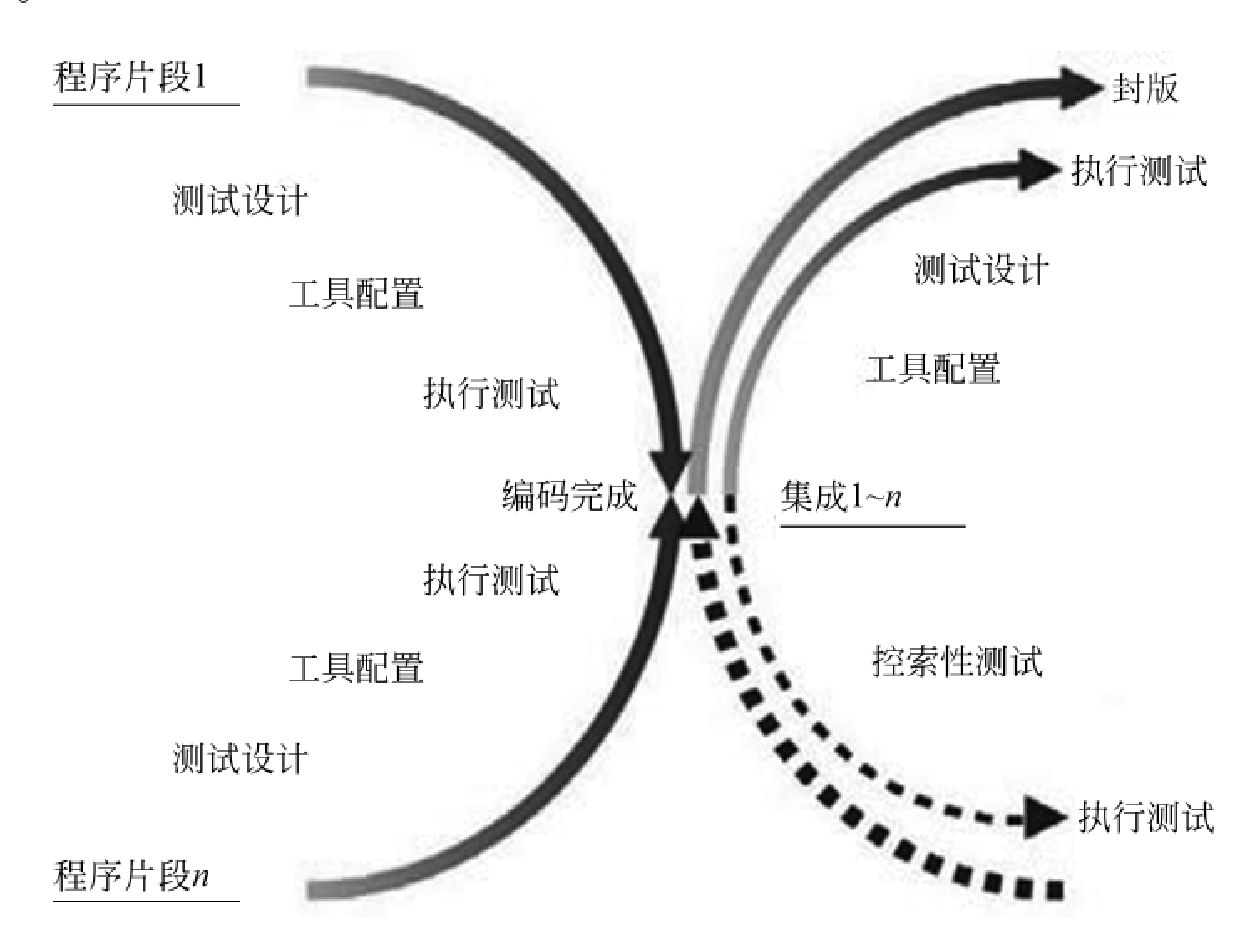


图 2.5　X 模型示意图

X 模型的左边描述单独的程序片段进行的编码和测试，通过集成成为可执行的程序。X 模型右上方为已通过集成测试的软件，也可以作为更大规模的集成的一部分，多条并行的曲线表示变更可以在各个部分发生。X 模型右下方定位了探索性测试，这是不进行事先计划的特殊测试，往往由有经验的测试人员执行。

5. 测试用例

测试用例是指对一项特定的软件产品进行测试的任务描述，体现为测试方案、方法、技术和策略等。测试用例的重要性体现在如下几个方面：

（1）测试用例构成了设计和制定测试过程的基础。

（2）测试工作量与测试用例的数量成比例。

（3）测试设计受控于测试用例。

（4）测试用例通常与测试类型或测试需求有关。

6. 测试停止标准

（1）软件系统经过单元测试、集成测试、系统测试，分别达到单元测试、集成测试、系统测试停止标准。

（2）软件系统通过验收测试，并已得出验收测试结论。

（3）软件项目需暂停以进行调整时，测试应随之暂停，并备份暂停点数据。

（4）软件项目在其开发生命周期内出现进度偏差，需暂停或终止时，测试应随之暂停或终止，并备份暂停点或终止点数据。

2.3 典型习题解析

2.3.1 选择题

1. 软件测试按照测试技术划分可分为(　　)。

A. 性能测试、负载测试、压力测试　　B. 恢复测试、安全测试、兼容测试

C. A 与 B　　D. 单元测试、集成测试、验收测试

【答案】 C

2. 软件测试的目的是(　　)。

A. 评价软件的质量　　B. 发现软件的错误

C. 找出软件中所有的错误　　D. 证明软件是正确的

【答案】 B

3. 下面(　　)属于动态分析。

A. 代码覆盖率　　B. 模块功能检查

C. 系统压力测试　　D. 程序数据流分析

【答案】 ABC

4. 下面(　　)属于静态分析。

A. 代码规则检查　　B. 程序结构分析
C. 程序复杂度分析　　D. 内存泄漏

【答案】 ABC

5. 若该模块已发现并改正的错误数目较多，则该模块中残留的错误与其他模块相比通常应该(　　)。

A. 较少　　B. 较多　　C. 相似　　D. 不确定

【答案】 B

6. 下面有关测试原则的说法正确的是(　　)。

A. 测试用例应由测试的输入数据和预期的输出结果两部分组成
B. 测试用例应选取合理的输入数据
C. 程序最好由编写该程序的程序员自己来测试
D. 使用测试用例进行测试是为了检查程序员是否做错了他该做的事

【答案】 A

7. 测试按照阶段划分为(　　)、集成测试、系统测试、验收测试(Alpha、Beta)，按照技术划分为(　　)、(　　)，按照性能划分为(　　)、压力测试、回归测试、恢复测试等。

A. 单元测试　　B. 黑盒测试　　C. 白盒测试　　D. 负载测试

【答案】 A，B，C，D

8. 代码走查和代码审查的主要区别是(　　)。

A. 在代码审查中由程序员来组织讨论，而在代码走查中由高级管理人员来领导评审小组的活动
B. 在代码审查中只检查代码是否有错误，而在代码走查中还要检查程序与设计文档的一致性
C. 在代码走查中只检查程序的正确性，而在代码审查中还要评审程序员的编程能力和工作业绩
D. 代码审查是一种正式的评审活动，而代码走查的讨论过程是非正式的

【答案】 D

9. 测试用例工作主要是(　　)。

A. 如何添加测试用例　　B. 如何编写测试用例
C. 将测试用例和需求关联　　D. 以上都对

【答案】 D

10. 软件测试用例主要由测试输入数据和(　　)两部分组成。

A. 测试计划　　B. 测试规则
C. 测试的预期结果　　D. 以往测试记录分析

【答案】 C

11. 软件测试按照测试技术划分为(　　)。

A. 性能测试、负载测试、压力测试　　B. 恢复测试、安全测试、兼容测试
C. A与B　　D. 单元测试、集成测试、验收测试

【答案】 C

12. 软件测试中常用的静态分析方法是(　　)。

①引用分析;②算法分析;③可靠性分析;④效率分析;⑤接口分析;⑥操作分析。

A. ①③　　B. ④⑥　　C. ②⑤　　D. ①⑤

【答案】 D

13. 软件测试过程模型有(　　)。

A. V 模型、H 模型　　B. W 模型

C. X 模型　　D. 以上都对

【答案】 D

14. 下面说法正确的是(　　)。

A. 程序测试无法确认程序没有错误

B. 黑盒测试是逻辑驱动的测试

C. 穷举测试一定可以暴露数据敏感错误

D. 白盒测试是一种输入输出驱动的测试

【答案】 A

15. 与设计测试数据无关的文档是(　　)。

A. 该软件的设计人员　　B. 程序的复杂程度

C. 源程序　　D. 项目开发计划

【答案】 D

16. 典型的软件测试过程模型有(　　)等。

A. V 模型、W 模型、H 模型、渐进模型

B. V 模型、W 模型、H 模型、螺旋模型

C. X 模型、W 模型、H 模型、前置测试模型

D. X 模型、W 模型、H 模型、增量模型

【答案】 C

17. 下列关于软件测试策略的叙述中不正确的是(　　)。

A. 增量测试的主要问题在于需要额外编写很多特殊的测试程序

B. 静态测试与动态测试都要执行程序

C. Myers 认为自底向上测试的方法要优于自顶向下测试的方法

D. 软件性能测试的目标之一是提高性能

【答案】 B

18. 测试程序时采用人工检查或计算机辅助静态分析的手段检查程序。这种测试称为(　　)。

A. 白盒测试　　B. 黑盒测试　　C. 静态测试　　D. 动态测试

【答案】 C

19. (　　)强调了测试计划等工作的先行和对系统需求和系统设计的测试。

A. V 模型　　B. W 模型　　C. 渐进模型　　D. 螺旋模型

【答案】 B

20. (　　)对软件测试流程给予了说明。

A. V 模型　　B. W 模型　　C. H 模型　　D. 增量模型

【答案】 C

21. A 模块中找到的错误最多,B 模块中找到的错误为平均水平,C 模块中找到的错误最少,则应该花费更多的时间和代价测试(　　)模块。

A. A　　B. B　　C. C　　D. A 和 C

【答案】 A

22. 在(　　)阶段中,发现并修复软件错误的代价更高。

A. 编码　　B. 单元测试　　C. 验收测试　　D. 运行

【答案】 D

23. 测试用例是测试使用的文档化的细则,其规定如何对软件某项功能或功能组合进行测试。测试用例应包括下列(　　)内容的详细信息。

①测试目标和被测功能；②测试环境和其他条件；③测试数据和测试步骤；④测试记录和测试结果。

A. ①③　　B. ①②③　　C. ①③④　　D. ①②③④

【答案】 B

24. 下面的描述中,不能体现前置测试模型要点的是(　　)。

A. 前置测试模型主张根据业务需求进行测试设计,认为需求分析阶段是测试计划与设计的最好时机

B. 前置测试模型将开发和测试的生命周期整合在一起,标识了项目生命周期从开始到结束之间的关键行为,提出业务需求最好在设计和开发之前就被正确定义

C. 前置测试将测试执行和开发结合在一起,并在开发阶段以编码—测试—编码—测试的方式来体现,强调对每一个交付的开发结果都必须通过一定的方式进行测试

D. 前置测试模型提出验收测试应该独立于技术测试,以保证设计及程序编码能够符合最终用户的需求

【答案】 A

25. 通常测试用例很难 100%覆盖测试需求,因为(　　)。

①输入量太大；②输出结果太多；③软件实现途径多；④测试依据没有统一标准。

A. ①②　　B. ①③　　C. ①②③　　D. ①②③④

【答案】 D

26. 造成测试覆盖率不达标的原因可能是(　　)。

A. 存在不可到达的代码或冗余的代码　　B. 测试用例不足

C. 存在不可能的路径和条件　　D. 以上全对

【答案】 D

2.3.2 判断题

1. 尽量用公共过程或子程序去代替重复的代码段。　　(×)

2. 测试是为了验证该软件已正确地实现了用户的要求。 (×)
3. 软件项目在进入需求分析阶段时,测试人员应该开始介入其中。 (√)
4. 验收测试是由最终用户来实施的。 (×)
5. 代码评审是检查源代码是否达到模块设计的要求。 (×)
6. 代码评审员一般由测试员担任。 (×)
7. 软件测试的目的是尽可能多地找出软件的缺陷。 (√)
8. 调试的目的是确定错误的位置和引起错误的原因,并加以改正。 (√)
9. 软件测试是执行程序,发现并排除程序中潜伏的错误的过程。 (√)
10. 动态测试有黑盒测试和白盒测试两种测试方法。 (√)
11. 软件验收测试包括 Alpha 测试和 Beta 测试。 (√)
12. 在软件测试的静态分析中进行人工测试的主要方法有代码审查和走查。 (√)
13. 测试是调试的一个部分。 (×)
14. 程序中隐藏错误的概率与其已发现的错误数成正比。 (√)
15. Beta 测试是验收测试的一种。 (×)
16. 测试人员要坚持原则,缺陷未修复完坚决不予通过。 (√)
17. 测试是证明软件正确的方法。 (×)
18. 集成测试计划在需求分析阶段末期提交。 (×)
19. 自底向上集成需要测试员编写驱动程序。 (√)
20. 负载测试是验证要检验的系统的能力最高能达到什么程度。 (√)
21. 测试中应该对有效和无效、期望和不期望的输入都要测试。 (√)
22. 对于连锁型分支结构,若有 n 个判定语句,则有 $2n$ 条路径。 (√)
23. 在某些情况下,保留 GOTO 语句反而能使写出的程序更加简洁。 (√)
24. 在没有产品说明书和需求文档的条件下可以进行动态黑盒测试。 (×)
25. 软件测试按照测试过程分为黑盒测试和白盒测试。 (×)
26. 错误发现得越迟,返工要做的事情就越多,成本就越高。 (√)

2.3.3 简答题

1. 软件测试的目的是什么?

【答】 软件测试的目的主要包括以下 3 点:

(1) 以最少的人力、物力、时间找出软件中潜在的各种错误和缺陷,通过修正错误和缺陷来提高软件质量,回避潜在的软件错误和缺陷给软件造成的商业风险。

(2) 通过分析测试过程中发现的问题可以帮助开发人员发现当前开发工作所采用的软件过程的缺陷,以便进行软件过程改进;同时通过对测试结果的分析整理,可以修正软件开发规则,并为软件可靠性分析提供相关的依据。

(3) 评价程序或系统的属性,对软件质量进行度量和评估,以验证软件的质量能够满足用户的需求,为用户选择、接收软件提供有力证据。

2. 软件测试中验证和确认有什么区别?

【答】 验证(verification)与确认(validation)看起来类似,但是它们的目标不同,处理的对象也不同。验证是检验软件工作产品是否符合规定的设计要求,而确认过程则要证明所开发的最终产品在其预定的环境中能发挥预定的作用,满足客户使用的需求。验证是以产品设计规格说明书作为依据的,而确认是以客户需求为目标的。

3. 软件测试的原则有哪些?

【答】 软件产品不同于一般的产品,有其自身独特之处,下面是软件测试的一些原则。

(1) 软件测试是证伪而非证真。

软件测试是为了发现错误而执行程序的过程,软件测试成功并不能说明软件不存在问题。

(2) 尽早地和不断地进行软件测试。

软件开发各个阶段工作的多样性以及参加开发各种层次人员之间工作的配合关系等因素使得开发的每个环节都可能产生错误。软件测试应在软件开发的需求分析和设计阶段就开始测试工作,编写相应的测试文档,坚持在软件开发的各个阶段进行技术评审和验证,这样才能尽早发现和预防错误,以较低的代价修改错误,提高软件质量。

(3) 重视无效数据和非预期的测试。

软件产品中暴露出来的许多问题常常是以某些非预期的方式运行时导致的。因此,测试用例的编写不仅应当考虑有效和正常的输入情况,而且也应当考虑无效和异常情况。

(4) 程序员应避免检查自己的程序。

人们具有一种不愿否定自己的心理本能,而这正是程序员不能检查自己的程序的原因。

(5) 充分注意测试中的群集现象。

经验表明,测试后程序中残存的错误数目与该程序中已发现的错误数目或检错率成正比。根据这个规律,若发现错误数目多,则残存错误数目也比较多,这就是错误群集现象。

(6) 用例要定期评审,适时补充修改用例

测试用例多次重复使用后,其发现缺陷的能力会逐渐降低。因此,测试用例需要进行定期评审和修改,不断增加新的不同的测试用例来发现潜在的更多的缺陷。

(7) 应当对每一个测试结果做全面检查。

不仔细全面地检查测试结果,就会使缺陷或错误被遗漏,因此,必须对预期的输出结果作出明确定义,对测试结果仔细分析检查。

(8) 测试现场保护和资料归档。

出现问题时要保护好现场,并记录足够的测试信息,以备缺陷能够复现。

4. 软件测试的覆盖率是什么?

【答】 测试覆盖率是评价测试活动覆盖产品代码的指标。测试的目的是确认产品代码按照预期一样工作,也可以看做是产品代码工作方式的说明文档。进一步考虑,测试覆盖率可以视为产品代码质量的间接指标。之所以说是间接指标,因为测试覆盖率评价的是测试代码的质量,并不是产品代码的质量。

5. V模型和W模型各自的优缺点是什么?

【答】 V模型反映了测试活动与开发活动的关系,标明了测试过程中存在的不同级别,并清楚地描述了测试的各个阶段和开发过程的各个阶段之间的对应关系。但是V模型仅把测试过程作为在需求分析、概要设计、详细设计及编码之后的一个阶段,主要针对程序进行寻找错误的活动,而忽视了测试活动对需求分析、系统设计等活动的验证和确认的功能。

相对于V模型而言,W模型增加了软件各开发阶段中应同步进行的验证和确认活动。W模型由两个V字形模型组成,分别代表测试与开发过程,明确表示出了测试与开发的并行关系。W模型有利于尽早地发现问题。但是,在W模型中,需求、设计、编码等活动被视为串行,测试和开发活动保持着一种线性的前后关系,上一阶段结束,才开始下一个阶段工作,因此,W模型无法支持迭代开发模型。

6. 动态测试和静态测试的区别是什么?

【答】 静态测试方法是指不运行被测程序本身,仅通过分析或检查源程序的语法、结构、过程、接口等来检查程序的正确性,对需求规格说明书、软件设计说明书、源程序做结构分析、流程图分析、符号执行来查找错误。静态测试方法通过程序静态特性的分析,找出欠缺和可疑之处,例如不匹配的参数、不适当的循环嵌套和分支嵌套、不允许的递归、未使用过的变量、空指针的引用和可疑的计算等。静态测试结果可用于进一步查错,并为测试用例选取提供指导。

动态测试方法是指通过运行被测程序,检查运行结果与预期结果的差异,并分析运行效率和健壮性等性能,这种方法由3部分组成:构造测试实例,执行程序,分析程序的输出结果。

7. 为什么需要测试用例?

【答】 如何以最少的资源投入,用最短的时间出色地完成测试,尽可能多地发现软件系统的缺陷,以提升软件的质量,是一个恒久不变的课题。影响软件测试的因素很多,例如需求定义的精确程度、软件本身的复杂度、系统设计的合理性以及开发人员的素质等,这些是开发层面的。而对于软件测试自身来说,不同的测试方法和技术的运用也会导致不一样的测试效果。如何才能保障软件测试质量的稳定呢?

一个有效方法就是基于测试用例的测试。测试用例(test case)简单来说就是预先编制的一组系统操作步骤和输入数据、执行条件以及预期结果,用以验证某个程序是否满足

某个特定需求。

8. 测试用例设计原则是什么？

【答】 设计测试用例时，应遵循以下原则：

（1）基于测试需求的原则。应按照测试类别的不同要求设计测试用例。例如，单元测试依据详细设计说明，集成测试依据概要设计说明，系统测试依据用户需求。

（2）基于测试方法的原则。应明确所采用的测试用例充分性要求，应采用相应的测试方法，如等价类划分、边界值分析、猜错法、因果图等方法。

（3）兼顾测试充分性和效率的原则。测试用例应兼顾测试的充分性和测试的效率。每个测试用例的内容也应完整，具有可操作性。

（4）测试执行的可再现性原则。应保证测试用例执行的可再现性，即对同样的测试用例，系统的执行结果应当是相同的。

9. 测试用例设计的步骤是什么？

【答】 软件测试用例的设计遵守如下 4 个步骤：

步骤 1，制定测试用例设计的策略和思想，在测试方案中描述出来。

步骤 2，设计测试用例的框架，也就是测试用例的结构。

步骤 3，细节结构，逐步设计出具体的测试用例。

步骤 4，通过测试用例的评审不断优化测试用例。

10. Beta 测试与验收测试是什么关系？

【答】 Beat 测试和验收测试是两种不同的测试。验收测试是以发现“未实现的需求”为目的，以评估“适合使用”为目标，而不是以发现缺陷为主要目的。Beta 测试是模拟真实的使用环境从而发现缺陷的一种测试。所以两者之间是非包容关系。

11. 测试用例设计的基本思想是什么？

【答】

（1）设计测试用例时，要寻求系统设计、功能设计的弱点。测试用例需要确切地反映功能设计中可能存在的各种问题，而不要简单地复制产品规格设计说明书的内容。

（2）设计正面的测试用例，应该参照设计规格说明书，根据关联的功能、操作路径等进行设计。而对孤立的功能则直接按功能设计测试用例。基本事件的测试用例应包含所有需要实现的需求功能，覆盖率达 100%。

（3）设计负面的、异常的测试用例，如考虑错误的或者异常的输入，往往可以发现更多的软件缺陷，这显得更为重要。例如，在进行电子邮件地址校验的时候，考虑错误的、不合法的（如没有@符号的输入）或者带有异常字符（单引号、斜杠、双引号等）的电子邮件地址输入，尤其是在做 Web 页面测试的时候，通常会出现因一些字符转义问题而造成的异常情况。

12. 什么是软件可测试性？

【答】 软件的可测试性是指软件发现故障并隔离、定位其故障的能力特性，以及在一定的时间和成本前提下进行测试设计、测试执行的能力。以下是一个常见的软件可测试性包括的内容：

- 可操作性——“运行得越好，测试的效率越高。”
- 可观察性——“所看见的就是所测试的。”
- 可控制性——“对软件的控制越好，测试越能够被自动执行与优化。”
- 可分解性——“通过控制测试范围，能够更好地分解问题，执行更灵巧的再测试。”
- 简单性——“需要测试的内容越少，测试的速度越快。”
- 稳定性——“改变越少，对测试的破坏越小。”
- 易理解性——“得到的信息越多，进行的测试越灵巧。”

13. 软件测试充分性准则有哪些？

【答】

(1) 空测试对任何软件都是不充分的。

(2) 对任何软件都存在有限的充分测试集合。

(3) 如果一个软件系统在一个测试数据集合上的测试是充分的，那么再多测试一些数据也应该是充分的。这一特性称为单调性。

(4) 即使对软件所有成分都进行了充分的测试，也并不意味着整个软件的测试已经充分了。这一特性称为非复合性。

(5) 即使对一个软件系统整体的测试是充分的，也并不意味着软件系统中各个成分都已经充分地得到了测试。这一特性称为非分解性。

(6) 软件测试的充分性应该与软件的需求和软件的实现都相关。

(7) 软件越复杂，需要的测试数据就越多。这一特性称为复杂性。

(8) 测试得越多，进一步测试所能得到的充分性增长就越少。这一特性称为回报递减率。



黑盒测试

3.1 本章要求

- 了解黑盒测试。
- 掌握等价类划分法。
- 掌握边界值分析法。
- 掌握因果图方法。
- 掌握决策表方法。
- 了解场景法。

3.2 本章知识重点

1. 黑盒测试

黑盒测试是从用户观点出发的测试，其目的是尽可能发现软件的外部行为错误。在已知软件产品功能的基础上，实现如下功能：

（1）检测软件功能能否按照需求规格说明书的规定正常工作，是否有功能遗漏。

（2）检测是否有人机交互错误，是否有数据结构和外部数据库访问错误，是否能恰当地接收数据并保持外部信息（如数据库或文件）等的完整性。

（3）检测行为、性能等特性是否满足要求等。

（4）检测程序初始化和终止方面的错误等。

黑盒测试具有如下两个显著的优点：

（1）黑盒测试与软件具体实现无关，所以如果软件实现发生了变化，测试用例仍然可以使用。

（2）设计黑盒测试用例可以和软件实现同时进行，因此可以压缩项目总的开发时间。

2. 黑盒测试方法

1）等价类划分法

等价类是指某个输入域的子集合。在该子集合中，测试某等价类的代表值就等于对这一类其他值的测试，对于揭露程序的错误是等效的。因此，将全部输入数据合理划分为

若干等价类,在每一个等价类中取一个数据作为测试的输入条件,就可以用少量有代表性的测试数据取得较好的测试结果。

2）边界值分析法

软件测试实践中,大量的错误往往发生在输入或输出范围的边界上,而不是发生在输入输出范围的内部。例如,数组下标、循环控制变量等边界附近往往出现大量错误。因此,边界值分析方法作为等价类划分方法的补充,不是选择等价类的任意元素,而是针对各种边界情况设计测试用例。

3）错误推测法

错误推测法是基于经验和直觉推测程序中所有可能存在的各种错误,从而有针对性地设计测试用例的方法。错误推测方法的基本思想是:列举出程序中所有可能有的错误和容易发生错误的特殊情况,根据它们选择测试用例。例如,在单元测试时曾列出的许多在模块中常见的错误,以前产品测试中曾经发现的错误等,这些就是经验的总结。又如,输入数据和输出数据为0。输入表格为空或输入表格只有一行,这些都是容易发生错误的情况。可选择这些情况下的例子作为测试用例。

4）决策表

决策表又称为判定表,是分析多种逻辑条件下执行不同操作的技术。在程序设计发展的初期,决策表是编写程序的辅助工具。决策表可以把复杂的逻辑关系和多种条件组合情况表达得更明确,与高级程序设计语言中的 if-else、switch-case 等分支结构语句类似,将条件判断与执行的动作联系起来。但与程序语言中的控制语句不同是,决策表能将多个独立的条件和多个动作联系清晰地表示出来。

5）因果图

等价类划分法和边界值分析法只是孤立地考虑各个输入数据的测试效果,没有考虑输入数据的组合及其相互制约关系。而输入条件的各种组合数目可能会是天文数字,因此必须考虑一种适合描述多种条件的组合,相应产生多个动作的方法,这就是因果图。

因果图利用图解法分析输入的各种组合情况,适合描述多种输入条件的组合,相应产生多个动作的方法。因果图具有如下好处:

(1) 考虑多个输入之间的相互组合、相互制约关系。

(2) 指导测试用例的选择,指出需求规格说明描述中存在的问题。

(3) 能够帮助测试人员按照一定的步骤高效率地开发测试用例。

(4) 因果图法是将自然语言规格说明转化成形式语言规格说明的一种严格的方法,可以指出规格说明存在的不完整性和二义性。

3.3 典型习题解析

3.3.1 选择题

1. 黑盒测试是通过软件的外部表现来发现软件缺陷的测试方法,包括(　　)等。

   A. 等价类划分法、因果图法、边界值分析法、错误推测法、判定表法

   B. 等价类划分法、因果图法、边界值分析法、正交试验法、符号法

C. 等价类划分法、因果图法、边界值分析法、功能图法、基本路径法
D. 等价类划分法、因果图法、边界值分析法、静态质量度量法、场景法

【答案】 A

2. 以下(　　)方法属于黑盒测试技术。

A. 基本路径测试　　B. 边界值分析测试
C. 循环覆盖测试　　D. 语句覆盖测试

【答案】 B

3. 下列方法中不属于黑盒测试的是(　　)。

A. 基本路径测试法　　B. 等价类
C. 边界值　　D. 场景法

【答案】 A

4. 黑盒测试方法的优点是(　　)。

A. 可测试软件的特定部位　　B. 能站在用户立场测试
C. 可按软件内部结构测试　　D. 可发现实现功能需求中的错误

【答案】 D

5. 对于业务流清晰的系统可以利用(①)贯穿整个测试用例设计过程;对于参数配置类的软件,要用(②)选择较少的组合方式达到最佳效果;如果程序的功能说明中含有输入条件的组合情况,则可以选用(③)和决策表法。

A. 等价类划分　　B. 因果图法　　C. 正交试验法　　D. 场景法
A. 等价类划分　　B. 因果图法　　C. 正交试验法　　D. 场景法
A. 等价类划分　　B. 因果图法　　C. 正交试验法　　D. 场景法

【答案】 ①D;②C;③B

6. (　　)方法根据输出对输入的依赖关系设计测试用例。

A. 路径测试　　B. 等价类　　C. 因果图　　D. 边界值

【答案】 C

7. 等价类划分完成后,得出(　　),它是确定测试用例的基础。

A. 有效等价类　　B. 无效等价类　　C. 等价类表　　D. 测试用例集

【答案】 C

8. 假设学生年龄的输入范围为16~40,则根据黑盒测试中的等价类划分技术,下面划分正确的是(　　)。

A. 可划分为2个有效等价类、2个无效等价类
B. 可划分为1个有效等价类、2个无效等价类
C. 可划分为2个有效等价类、1个无效等价类
D. 可划分为1个有效等价类、1个无效等价类

【答案】 B

9. 在黑盒测试中,着重检查输入条件的组合的测试用例设计方法是(　　)。

A. 等价类划分法　　B. 边界值分析法
C. 错误推测法　　D. 因果图法

【答案】 D

10. 除了测试程序外，黑盒测试还适用于对(　　)阶段的软件文档进行测试。

A. 编码　　B. 软件详细设计
C. 软件总体设计　　D. 需求分析

【答案】 D

11. 由因果图转换而来的(　　)是确定测试用例的基础。

A. 决策表　　B. 约束条件表　　C. 输入状态表　　D. 输出状态表

【答案】 A

12. (　　)适合检查程序输入条件的各种组合情况。使用该方法首先要理解软件所表示的对象及其关系。然后，定义一组保证“所有对象与其他对象都具有所期望的关系”的测试序列。

A. 等价类划分法　　B. 边界值分析法
C. 因果图法　　D. 决策表法

【答案】 C

13. 黑盒测试是一种重要的测试策略，又称为数据驱动的测试，其测试数据来源于(　　)。

A. 软件规格说明　　B. 软件设计说明
C. 概要设计说明　　D. 详细设计说明

【答案】 A

14. 在设计测试用例时，(　　)是用得最多的一种黑盒测试方法。

A. 等价类划分法　　B. 边界值分析法
C. 因果图法　　D. 功能图法

【答案】 B

15. 下面为一段 C 语言程序，边界值问题可以定位在(　　)。

```
int data(3)
int i
for(i=1,i<=3,i++)
    data(i)=100
```

A. data(0)　　B. data(1)　　C. data(2)　　D. data(3)

【答案】 A

16. 假定 X 为整数类型变量，X≥1 并且 X≤10，如果用边界值分析法，X 在测试中应该取(　)值。

A. 1,10　　B. 0,1,10,11
C. 1,11　　D. 1,5,10,11

【答案】 B

17. 用黑盒技术设计测试用例的方法之一为(　　)。

A. 因果图　　B. 逻辑覆盖　　C. 循环覆盖　　D. 基本路径测试

【答案】 A

18. 在边界值分析中，下列数据通常不用来做数据测试的是（　　）。

A. 正好等于边界的值　　B. 等价类中的等价值

C. 刚刚大于边界的值　　D. 刚刚小于边界的值

【答案】 B

3.3.2 判断题

1. 黑盒测试时，测试用例是根据程序内部逻辑设计的。（×）
2. 黑盒测试方法中最有效的是因果图法。（√）
3. 尽量采用复合的条件测试，以避免嵌套的分支结构。（√）
4. 黑盒测试也称为结构测试。（×）

3.3.3 简答题

1. 什么是黑盒测试方法？

【答】 黑盒测试方法是把程序看作一个不能打开的黑盒子，不考虑程序内部结构和内部特性，而是考察数据的输入、条件限制和数据输出，完成测试。黑盒测试一般须在完成集成测试后进行，而且是针对应用系统进行测试。功能测试是基于产品功能说明书，是在已知产品所应具有的功能前提下，从用户角度来进行功能验证，以确认每个功能是否都能正常使用的方法，包括等价类划分法、边界值分析法、因果图法、决策表法等。

2. 什么是等价类划分？

【答】 等价类是某个输入域的子集，在该子集中每个输入数据的作用是等效的。等价类划分法将程序可能的输入数据分成若干个子集，从每个子集选取一个有代表性的数据作为测试用例，在分析需求规格说明的基础上划分等价类，列出等价类表。等价类分为有效等价类和无效等价类。其中，有效等价类是有意义的、合理的输入数据，可以检查程序是否实现了规格说明中所规定的功能和性能；而无效等价类和有效等价类相反，即不满足程序输入要求或者无效的输入数据构成的集合。

3. 边界值法设计遵循的原则是什么？

【答】 边界值法设计遵循的原则如表 3.1 所示。

表 3.1 边界值法设计遵循的原则

输入条件（数据）	输入边界值数据
规定了取值范围	刚刚达到这个范围，刚刚超越这个范围
规定值的个数	最大个数，比最大个数大 1，最小个数，比最小个数少 1
输入或输出是一个有序集合	集合的第一个、最后一个元素
程序中使用一个内部数据结构	内部数据结构边界上的值

4. 边界值分析的设计原则是什么？

【答】 边界值分析法是确定边界情况（输入或输出等价类的边界），选取正好等于、刚

刚大于或刚刚小于边界值作为测试数据。边界值分析测试用例的获得方法是：使一个变量取极值(最小值、略高于最小值、正常值、略低于最大值、最大值)，使所有其余的变量取正常值，边界值的数目为 $4n+1$；而健壮性测试的数目为 $6n+1$，即再增加一个略小于最小值(min－)、一个略大于最大值(max＋)。

5. 等价类划分法、边界值分析法、决策表法之间的关系是什么？

【答】 等价类划分法是通过等价类划分减少测试用例的绝对数量，适用于强数据类型编程语言，但等价类划分法只是机械地从对应等价类中选择输入值而不考虑其应用领域的相关知识。

边界值分析法适合当被测程序含有多个独立变量的函数，而且这些变量受物理量的限制的情况，对布尔变量和逻辑变量没有多大意义。

决策表法通过分析被测程序的逻辑依赖关系构造决策表，设计测试用例。

等价类划分法、边界值分析法和决策表法生成测试用例的数量与开发测试用例所需工作量的对比如图 3.1 所示。

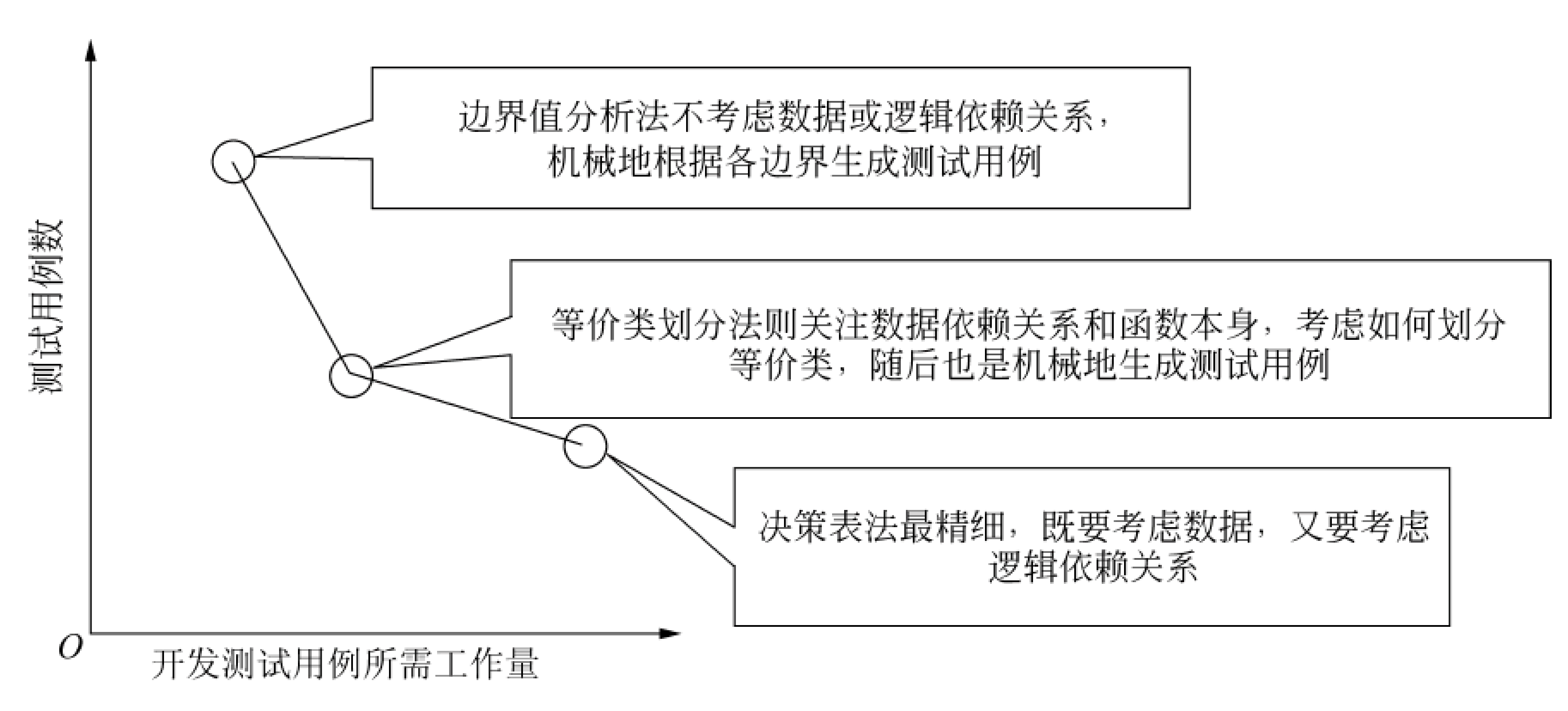


图 3.1　3 种黑盒测试法测试用例数与开发测试用例所需工作量对比图

6. 黑盒测试用例设计方法的策略是什么？

【答】

(1) 首先进行等价类划分，包括输入条件和输出条件的等价类划分，将无限测试变成有限测试，这是减少测试量和提高测试效率最有效的办法。

(2) 在任何情况下都必须使用边界值分析方法。此方法设计的测试用例发现程序错误的能力最强。

(3) 可以用错误和推测法追加一些测试用例。

(4) 对照程序的逻辑，检查已设计的测试用例的逻辑覆盖度，如果没有达到要求，应再补充。

(5) 如果程序的功能说明中含有输入条件的组合情况，一开始就可以使用因果图法和决策表法。

(6) 对于参数配置类的软件,要用正交试验法选择较少的组合方式达到最佳效果。

(7) 功能图法也是很好的测试用例设计方法,可以通过不同时期条件的有效性设计不同的数据。

(8) 对于业务流清晰的系统,可以利用场景法贯穿整个测试案例过程,在案例中综合使用各种方法。

7. 什么是场景法和正交实验法?

【答】 场景法基于用例测试场景,测试用例的设计业务路径,从用例开始到结束遍历其中所有基本流和备选流。正交实验法是指从大量的试验数据中挑选适量的有代表性的点,从而合理地安排测试的一种科学试验设计方法。

8. 因果图法相对于等价类划分法和边界值分析法有什么优点?

【答】 因果图法是利用图解法分析输入的各种组合情况,从而设计测试用例的方法,它适合检查程序输入条件的各种组合情况。等价类划分法和边界值分析法都是着重考虑输入条件,但没有考虑输入条件的组合、输入条件之间相互制约的关系。这样虽然将各种输入条件可能出错的情况已经测试到了,但多个输入条件组合起来可能出错的情况却被忽视了。如果在测试时必须考虑输入条件的各种组合,则可能的组合数目将是天文数字,因此必须考虑采用一种适合于描述多种条件的组合,相应产生多个动作的形式来进行测试用例的设计,这就需要使用因果图法。

3.3.4 设计题

1. 输入框要求:"用户名由字母开头,后跟字母或数字的任意组合构成。有效字符数不超过8个。"

要求设计出有效等价类和无效等价类。

【解析】 等级类划分如表3.2所示。

表3.2 等价类划分

输入条件	有效等价类	无效等价类
第一个字符	字母(1)	数字(2)
标示符字符数	1~8个(3)	0个(4),8个(5)
标示符组成	字母(6),数字(7)	非字母数字字符(8),保留字(9)

2. 采用等价类划分法设计三角形问题的测试用例。

【解析】 分析题目中给出和隐含的对输入条件的要求如下:

①整数;②三个数;③非零数;④正数;⑤两边之和大于第三边;⑥等腰;⑦等边。

如果a、b、c满足条件(①~④),则输出下列4种情况之一:

(1) 如果不满足条件⑤,则程序输出为"非三角形"。

(2) 如果3条边相等即满足条件⑦,则程序输出为"等边三角形"。

(3) 如果只有两条边相等,即满足条件⑥,则程序输出为"等腰三角形"。

(4) 如果3条边都不相等,则程序输出为"一般三角形"。

划分等价类并编号,如表3.3所示。

表 3.3 等价类划分

<table>
<tr><th colspan="2">输入和输出</th><th>有效等价类</th><th>无效等价类</th></tr>
<tr><td rowspan="4">输入条件</td><td rowspan="4">输入3个整数</td><td>整数(1)</td><td>a为非整数(12)
b为非整数(13)
c为非整数(14)
a、b为非整数(15)
b、c为非整数(16)
c、a为非整数(17)
a、b、c为非整数(18)</td></tr>
<tr><td>3个数(2)</td><td>只有a(19)
只有b(20)
只有c(21)
只有a、b(22)
只有b、c(23)
只有c、a(24)
3个数a、b、c(25)</td></tr>
<tr><td>非零数(3)</td><td>a为0(26)
b为0(27)
c为0(28)
a、b为0(29)
b、c为0(30)
c、a为0(31)
3个数a、b、c为0(32)</td></tr>
<tr><td>正数(4)</td><td>a<0(33)
b<0(34)
c<0(35)
a<0且b<0(36)
b<0且c<0(37)
c<0且a<0(38)
a<0且b<0且c<0(39)</td></tr>
<tr><td rowspan="3">输出条件</td><td>构成一般三角形</td><td>a+b>c(5)
b+c>a(6)
a+c>b(7)</td><td>a+b=c(40)
a+b<c(41)
b+c=a(42)
b+c<a(43)
a+c=b(44)
a+c<b(45)</td></tr>
<tr><td>构成等腰三角形</td><td>a=b(8)
b=c(9)
a=c(10)</td><td></td></tr>
<tr><td>构成等边三角形</td><td>a=b=c(11)</td><td></td></tr>
</table>

覆盖有效等价类的测试用例如表3.4所示。

表 3.4 有效等价类的测试用例

a b c	覆盖等价类号码
3 4 5	(1)～(7)
4 4 5	(1)～(8)
4 5 5	(1)～(7),(9)
5 4 5	(1)～(7),(10)
4 4 4	(1)～(7),(11)

覆盖无效等价类的测试用例如表 3.5 所示。

表 3.5 无效等价类的测试用例

a	b	c	覆盖等价类
2.5	4	5	(12)
3	4.5	5	(13)
3	4	5.5	(14)
3.5	4.5	5	(15)
3	4.5	5.5	(16)
3.5	4	5.5	(17)
4.5	4.5	5.5	(18)
3			(19)
	4		(20)
		5	(21)
3	4		(22)
	4	5	(23)
3		5	(24)
3	4	5	(25)
0	4	5	(26)
3	0	5	(27)
3	4	0	(28)
0	0	5	(29)
3	0	0	(30)
0	4	0	(31)
0	0	0	(32)
−3	4	5	(33)
3	−4	−5	(34)
3	4	−5	(35)
−3	−4	5	(36)
−3	4	−5	(37)
3	−4	−5	(38)
−3	−4	−5	(39)

续表

a	b	c	覆盖等价类
3	1	5	(40)
3	2	5	(41)
3	1	1	(42)
3	2	1	(43)
1	4	2	(44)
3	4	1	(45)

3. 每个学生可以选修 1～3 门课程,要求采用等价类设计测试用例。

【解析】

步骤 1：等级类划分如表 3.6 所示。

表 3.6　等价类划分

输入条件	有效等价类	无效等价类
选修课程	选修 1 门(1)	没选课(2) 选 3 门以上(3)

步骤 2：根据等级类划分设计测试用例,如表 3.7 所示。

表 3.7　测试用例

选修课程	覆盖等价类号码
选修＝2	(1)
选修＝0	(2)
选修＞3	(3)

4. 系统要求用户输入以年月表示的日期。假设日期限定在 1990 年 1 月至 2049 年 12 月,并规定日期由 6 位数字字符组成,前 4 位表示年,后 2 位表示月。现用等价类划分法设计测试用例(不考虑 2 月的问题)。

【解析】

步骤 1：划分等价类并编号,如表 3.8 所示。

表 3.8　等价类划分

输入等价类	有效等价类	无效等价类
日期的类型及长度	6 位数字字符(1)	有非数字字符(2) 少于 6 位数字字符(3) 多于 6 位数字字符(4)
年份范围	1990～2049(5)	小于 1990(6) 大于 2049(7)
月份范围	01～12(8)	等于 00(9) 大于 12(10)

步骤 2：针对有效等价类设计测试用例和结果，如表 3.9 所示。

表 3.9 有效等价类测试用例

测试数据	期望结果	覆盖的有效等价类
200211	输入有效	(1)(5)(8)

针对无效等价类设计测试用例和结果，如表 3.10 所示。

表 3.10 无效等价类测试用例

测试数据	期望结果	覆盖的无效等价类
95June	无效输入	(2)
20036	无效输入	(3)
2001006	无效输入	(4)
198912	无效输入	(6)
200401	无效输入	(7)
200100	无效输入	(9)
200113	无效输入	(10)

5. 以 0x 或 0X 开头的十六进制整数，其取值范围为－7f～7f(a～f 不区分大小写字母)，如 0x13、0x6A、－0x3c。请采用等价类划分的方法设计测试用例。

【解析】

步骤 1：等价类划分如表 3.11 所示。

表 3.11 等价类划分

输入条件	有效等价类	无效等价类
开头字符	由 0x 或 0X 开头(1)	以字母开头(2)，以非 0 数字开头(3)
数值字符	数字或 A～F 的字母(4)	A～F 以外的字母(5)
数值字符个数	≥1 个(6)	0 个(7)
数值	≥－7f 且≤7f(8)	<－7f(9) >7f(10)

步骤 2：设计测试用例，如表 3.12 所示。

表 3.12 测试用例

序　号	测试数据	覆盖等价类
1	0x7F	(1)(4)(6)(8)
2	－0Xb	(1)(4)(6)(8)
3	0X0	(1)(4)(6)(8)

续表

序　号	测试数据	覆盖等价类
4	0x	(1)(7)
5	A7	(2)
6	－1A	(3)
7	0X8h	(1)(5)
8	0x80	(1)(4)(10)
9	－0XaB	(1)(4)(9)

6. 设计决策表表达读课文的情况。

【解析】 决策表如表 3.13 所示。

表 3.13　决策表

条件桩		动作桩							
		1	2	3	4	5	6	7	8
问题	觉得疲倦吗?	Y	Y	Y	Y	N	N	N	N
	感兴趣吗?	Y	Y	N	N	Y	Y	N	N
	糊涂吗?	Y	N	Y	N	Y	N	Y	N
建议	重读					√			
	继续						√		
	跳下一章							√	√
	休息	√	√	√	√				

7. 采用边界值分析法设计三角形问题的测试用例(三角形的三边 a、b、c 取值区间为1～100)。

【解析】 一般性边界值测试用例如表 3.14 所示。

表 3.14　一般性边界值测试用例

测试用例	a	b	c	预期输出
Test1	1	50	50	等腰三角形
Test2	2	50	50	等腰三角形
Test3	99	50	50	等腰三角形
Test4	100	50	50	非三角形
Test5	50	1	50	等腰三角形
Test6	50	2	50	等腰三角形
Test7	50	99	50	等腰三角形

续表

测试用例	a	b	c	预期输出
Test8	50	100	50	非三角形
Test9	50	50	1	等腰三角形
Test10	50	50	2	等腰三角形
Test11	50	50	99	等腰三角形
Test12	50	50	100	非三角形
Test13	50	50	50	等边三角形

健壮性边界值测试用例如表 3.15 所示。

表 3.15　健壮性边界值测试用例

测试用例	a	b	c	预期输出
Test1	0	50	50	无效输入
Test2	1	50	50	等腰三角形
Test3	2	50	50	等腰三角形
Test4	99	50	50	等腰三角形
Test5	100	50	50	非三角形
Test6	101	50	50	无效输入
Test7	50	0	50	无效输入
Test8	50	1	50	等腰三角形
Test9	50	2	50	等腰三角形
Test10	50	99	50	等腰三角形
Test11	50	100	50	非三角形
Test12	50	101	50	无效输入
Test13	50	0	0	无效输入
Test14	50	50	1	等腰三角形
Test15	50	50	2	等腰三角形
Test16	50	50	99	等腰三角形
Test17	50	50	100	非三角形
Test18	50	50	101	无效输入
Test19	50	50	50	等边三角形

8. 某一软件项目的规格说明：对于处于提交审批状态的单据，数据完整率达到 80% 以上或已经过业务员确认，则进行处理。采用基于因果图的方法为该软件项目设计决

策表。

【解析】 首先根据程序的规格说明，对于所有可能的输入和输出条件，找出所有的原因和结果以及二者之间的关系，画出因果图。然后基于因果图的方法设计测试用例。

步骤1：首先根据规格说明列出所有可能的输入和输出，得到如下结果。

- 输入：处于提交状态，数据完整率达到80%以上，已经过业务员确认。
- 输出：处理或不处理。

找出所有输入与输出的关系，通过分析，得到以下的对应关系：

- 如果单据处于提交审批状态且数据完整率达到80%以上，则处理。
- 如果单据不处于提交审批状态，则不处理。
- 如果单据处于提交审批状态，数据完整率未达到80%以上，但已经过业务员确认，则处理。

下面列出所有的原因和结果，并进行编号。

原因：1—处于提交状态。

2—数据完整率未达到80%以上。

3—已经过业务员确认。

结果：21—处理。

22—不处理。

步骤2：根据根据上面分析的关系，画出因果图，如图3.2所示。

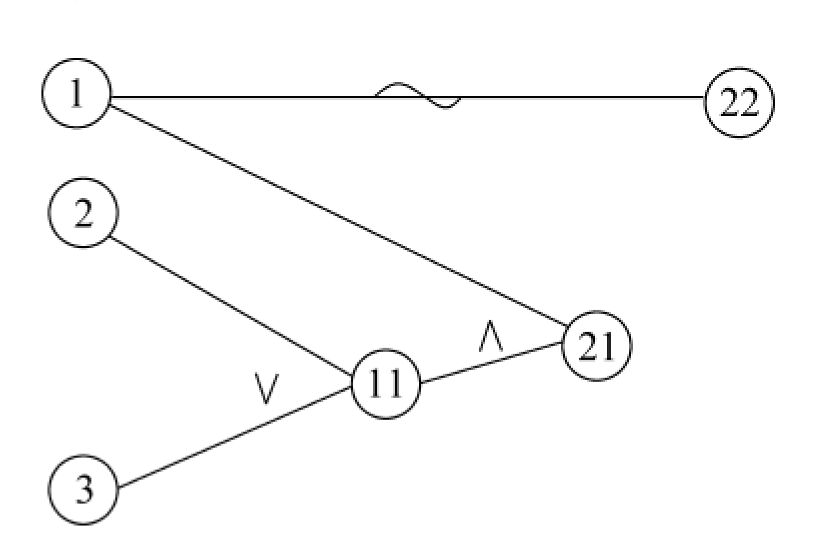


图3.2 因果图

步骤3：将因果图转换成决策表，如表3.16所示。

表3.16 决策表

条件桩		动作桩							
		1	2	3	4	5	6	7	8
条件	1	Y	Y	Y	Y	N	N	N	N
	2	Y	Y	N	N	Y	Y	N	N
	3	Y	N	Y	N	Y	N	Y	N
中间结果	11	Y	Y	Y	N	Y	Y	Y	N
动作	21	Y	Y	Y	N	N	N	N	N
	22	N	N	N	Y	Y	Y	Y	Y

9. 假设有一个把数字串转换为整数的函数。其中数字串要求长度为1～6,机器字长为16位。分析程序中出现的边界情况,采用边界值分析法为该程序设计测试用例。

【解析】 分析该程序的说明和功能,显然可以划分4组测试等价类:有效输入等价类、无效输入等价类、合法输出等价类和非法输出等价类。在考虑该程序的合法输出和非法输出时需要考虑计算机的字长,这时要使用边界值分析法设计测试用例,以补充等价类法设计的测试用例。具体采用边界值分析法设计的测试用例如下:

(1) 使程序输入刚好等于最小的负整数。

输入:'—32768'

输出:—32768

(2) 使程序输入刚好等于最大的正整数。

输入:'32767'

输出:32767

(3) 使程序输入刚好小于最小的负整数。

输入:'—32769'

输出:错误

(4) 使程序输入刚好大于最大的正整数。

输入:'2768'

输出:错误

10. 一个售货机软件,若投入1元币,按下"可乐""雪碧"或"红茶"按钮,送出相应的饮料;若投入的是2元币,在送出饮料的同时退还1元币。要求设计出决策表。

【解析】 当有投币和按按钮动作时,就会有相应的饮料送出;若投币为2元,除了送出饮料,还会退1元硬币;只有按按钮或只有投币动作时,就不会有输出结果。决策表如表3.17所示。

表3.17 决策表

测试用例		1	2	3	4	5
输入	投入1元币	1	1	0	0	0
	投入2元币	0	0	1	0	0
	按"可乐"按钮	1	0	0	0	0
	按"雪碧"按钮	0	0	0	1	0
	按"红茶"按钮	0	0	1	0	1
输出	退还1元币	0	0	1	0	0
	送出"可乐"饮料	1	0	0	0	0
	送出"雪碧"饮料	0	0	0	0	0
	送出"红茶"饮料	0	0	1	0	0

表中1表示执行该动作,0表示不执行该动作。

11. 电力公司把用户分为单费率用户和复费率用户两类。对于单费率用户实行单一电价,即在任何时间段都是一个电价;对于复费率用户,在不同时间段实行不同的电价。有如下4条计算电费的规则:

(1) 对于单费率用户,按公式A计算电费。

(2) 对于复费率用户,如果不在规定时间段内,同单费率用户,按照公式A计算电费。

(3) 对于复费率用户,如果在规定时间段内,同单费率用户,按照公式B计算电费。

(4) 如果既不是单费率用户也不是复费率用户,则做其他处理。

要求采用决策表设计测试用例,尽量使其覆盖所有的情况。

【解析】 表3-18为由电费计算的决策表导出的测试用例。

表3.18 决策表导出的测试用例

测试用例	输入条件	预期结果
1	单费率用户	按公式A计算电费
2	复费率用户,不在规定时间段内	按公式A计算电费
3	复费率用户,在规定期时间段内	按公式B计算电费
4	其他用户	做其他处理

12. 用因果图设计"中国象棋中走马"的测试用例。

【解析】

步骤1:分析中国象棋中走马的实际情况。

情况1:如果落点在棋盘外,则不移动棋子。

情况2:如果落点与起点不构成日字形,则不移动棋子。

情况3:如果落点处有自己方棋子,则不移动棋子。

情况4:如果在落点方向的邻近交叉点有棋子(绊马腿),则不移动棋子。

情况5:如果不属于情况1到情况4,且落点处无棋子,则移动棋子。

情况6:如果不属于情况1到情况4,且落点处为对方棋子(非老将),则移动棋子并除去对方棋子。

情况7:如果不属于情况1到情况4,且落点处为对方老将,则移动棋子,并提示战胜对方,游戏结束。

步骤2:根据分析明确原因和结果。

原因如下:

1—落点在棋盘上。

2—落点与起点构成日字形。

3—落点处为自己方棋子。

4—落点方向的邻近交叉点无棋子。

5—落点处无棋子。

6—落点处为对方棋子(非老将)。

7—落点处为对方老将。

注意：第 4 点落点方向的邻近交叉点有棋子，说明不能移动棋子，其他原因都可以移动棋子。

结果如下：

21—不移动棋子。

22—移动棋子。

23—移动棋子，并除去对方棋子。

24—移动棋子，并提示战胜对方，结束游戏。

注意：在移动棋子的时候，由于有 3 种结果，所以要增加一个中间节点，表明过渡状态。添加中间节点 11，目的是作为导出结果的进一步原因，简化因果图导出的决策表。

因果图如图 3.3 所示。

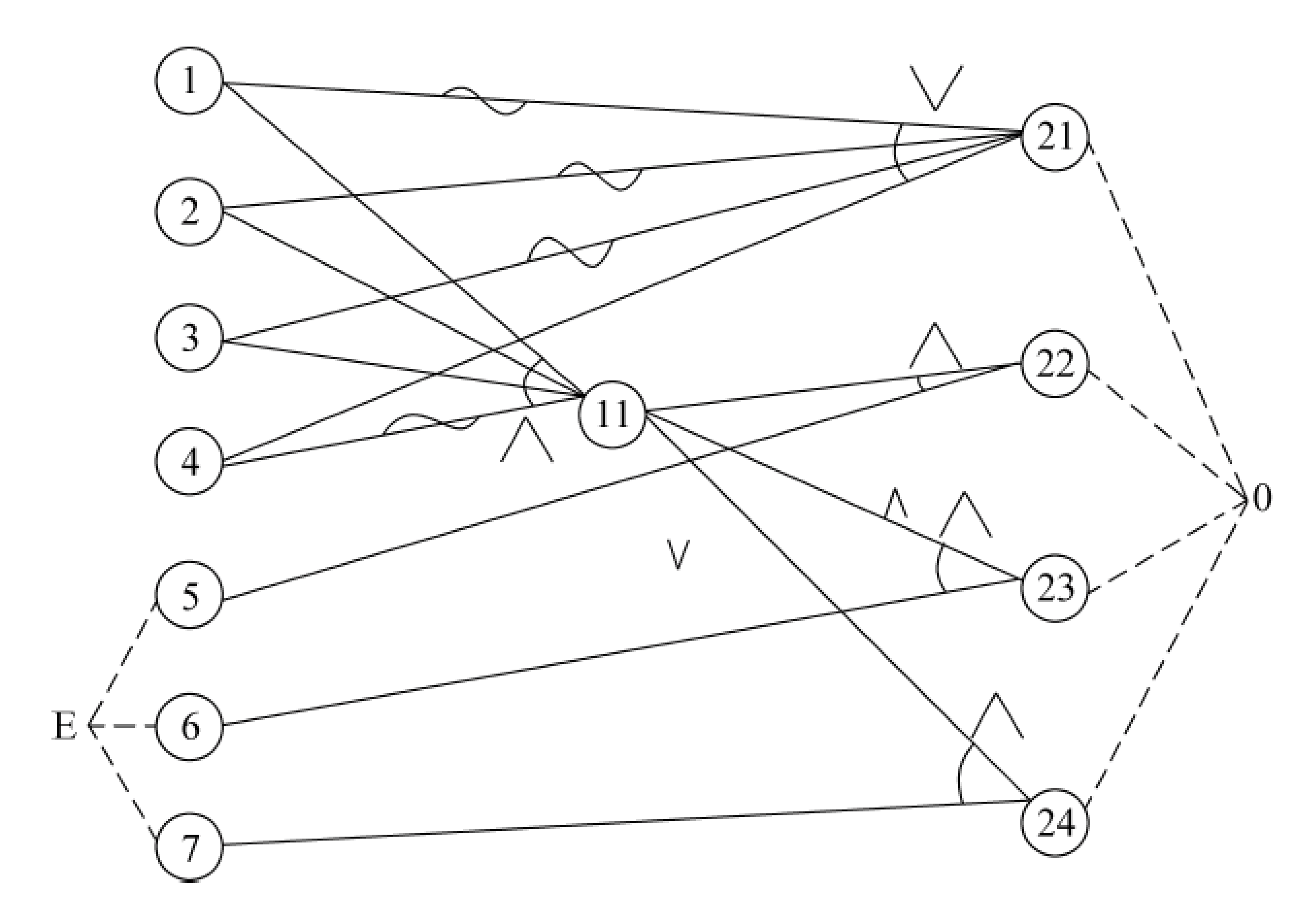


图 3.3　因果图

由因果图推理出表 3.19 所示的表。

表 3.19　决策表

条 件 桩		动 作 桩							
		1	2	3	4	5	6	7	8
原因	1	1	1	1	1	0	0	0	0
	2	1	1	0	0	1	1	0	0
	3	1	0	1	0	1	0	1	0
	11			1	1	1	1	0	0
结果	22			0	0	0	0	1	1
	21			1	0	1	0	0	0
	23			0	1	0	1	0	1
测试用例				A3 A8	AB A?	B5 B4	BN B!	C2 X6	SD P$

原因 5、6、7 不能同时发生，所以对其施加异约束 E，考虑结果不能同时发生，施加唯一约束 O。

步骤 3：根据因果图建立决策表，如表 3.20 和表 3.21 所示。

表 3.20　决策表 1

条件桩		动作桩															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
原因	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
结果	11	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	21	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
用例																	

表 3.21　决策表 2

条件桩		动作桩															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
原因	11	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	5	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	6	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	7	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
结果	22	0		0	1	0	0			0	0						
	23	0		0	0	0	1			0	0						
	24	0		0	0	0	0			0	1						
用例																	

注意：

（1）决策表中，由于表格大小限制，没有列出最后所选的测试用例。

（2）表 3.21 中部分列为空白，表示不可能发生的现象。

（3）通过中间节点将用例的决策表简化为两个表以减少工作量。

13. 某保险公司保费计算公式为：保费＝投保额×保险费率。其中，保险费率根据年龄、性别、婚姻状况和抚养人数的不同而不同，具体规则见表 3.22。

要求通过对规格说明输入数据的取值分析，设计保险公司人寿保险保费计算程序的等价类。

表 3.22 保险费具体规则

输入条件	年　龄			性　别		婚姻状况		抚养人数	
取值	20～39	40～59	其他	M	F	已婚	未婚	1～5 人	6～9 人
费率	6 点	4 点	2 点	4 点	3 点	3 点	5 点	3 点	5 点

【解析】

步骤 1：分析程序规格说明中输入条件的要求。

- 年龄：数字，取值范围为 1～99，必填。
- 性别：一位英文字符，只能取 M 或 F，必填。
- 婚姻：字符，只能取“已婚”或“未婚”，必填。
- 抚养人数：数字，取值范围为 1～9，选填。

对每个输入条件进行等价类划分，如表 3.23 所示。

表 3.23 等价类划分

输入条件	有效等价类	编号	无效等价类	编号
年龄	20～39 岁	(1)		
	40～59 岁	(2)		
	1～19 岁 60～99 岁	(3)	小于 1	(12)
			大于 99	(13)
性别	单个英文字符	(4)	非英文字符	(14)
			非单个英文字符	(15)
	M	(5)	除 M 和 F 之外的其他单个字符	(16)
	F	(6)		
婚姻	已婚	(7)	除“已婚”和“未婚”之外的其他字符	(17)
	未婚	(8)		
抚养人数	空白	(9)	除空白和数字之外的其他字符	(18)
	1～5 人	(10)	小于 1	(19)
	6～9 人	(11)	大于 9	(20)

步骤 2：根据等价类划分，设计能覆盖所有等价类的测试用例，如表 3.24 所示。

表 3.24 测试用例

测试用例编号	输入数据				预期输出	覆盖等价类
	年龄	性别	婚姻	抚养人数		
1	27	F	未婚	空白	14 点	(1)(6)(8)(9)
2	50	M	已婚	1～6	14 点	(2)(5)(7)(10)

续表

测试用例编号	输入数据				预期输出	覆盖等价类
	年龄	性别	婚姻	抚养人数		
3	70	F	已婚	6～9	13点	(3)(6)(7)(11)
4	0	M	未婚	空白	提示年龄错误	(12)(5)(8)(9)
5	100	F	已婚	1～5	提示年龄错误	(13)(3)(7)(10)
6	99	男	已婚	6～9	提示性别错误	(3)(14)(7)(10)
7	1	child	未婚	空白	提示性别错误	(3)(15)(8)(9)
8	45	N	已婚	1～5	提示性别错误	(2)(16)(7)(10)
9	38	F	离婚	6～9	提示婚姻错误	(1)(6)(17)(11)
10	62	M	已婚	没有	提示抚养人数错误	(3)(5)(7)(18)
11	18	F	未婚	0	提示抚养人数错误	(3)(6)(8)(19)
12	40	M	未婚	10	提示抚养人数错误	(2)(5)(8)(20)



白 盒 测 试

4.1　本 章 要 求

- 了解白盒测试发展历程。
- 掌握逻辑覆盖法的各种方法。
- 了解路径分析与测试。
- 掌握数据流测试。
- 了解白盒测试综合策略。
- 掌握 3 种调试技术。

4.2　本章知识重点

1. 白盒测试概述

白盒测试是对软件的过程性细节做细致的检查，把测试对象看作一个打开的盒子，它允许测试人员利用程序内部的逻辑结构及有关信息设计或选择测试用例，对程序所有逻辑路径进行测试。通过在不同点检查程序状态，确定实际状态是否与预期的状态一致。因此白盒测试又称为结构测试或逻辑驱动测试。

2. 静态白盒测试

静态白盒测试是指计算机不真正运行被测试的程序，通过人工方式对程序和文档进行分析与检查，主要检查代码和设计的一致性、代码对文档标准的遵循及代码的可读性、逻辑表达正确性、结构的合理性等方面。代码检查一般在编译和动态测试之前进行，包括走查、审查（或评审）、伙伴检查等，如表 4.1 所示。

表 4.1　走查、审查、伙伴检查的对比

事　项	走　　查	审　　查	伙 伴 检 查
准备	通读设计和编码	准备好需求描述文档、程序设计文档、程序的源代码清单、代码编码标准和代码缺陷检查表	没有准备

续表

事　项	走　　查	审　　查	伙伴检查
形式	非正式会议	正式会议	
主持人	任何人	由非该软件的编制人员组成	没有
参加人员	开发人员为主,2～7人小组	3～6人小组,项目组成员,包括测试人员	1～2人
主要技术	无	缺陷检查表	无
注意事项	限时,不要现场修改代码	限时,不要现场修改代码	无
生成文档	会议记录	静态分析错误报告	口头评论
目标	代码标准规范,无逻辑错误	代码标准规范,无逻辑错误	无
优点	能使更多人熟悉产品	费用低	费用低
缺点	查出的故障较少	短期成本高	查出的故障较少

3. 逻辑覆盖法

逻辑覆盖方法又称为控制流覆盖,设计测试用例满足如下覆盖标准:语句覆盖、判定覆盖、条件覆盖、判定-条件覆盖、条件组合覆盖和路径覆盖等。

1）语句覆盖

语句覆盖又称为线覆盖面或段覆盖面。其含义是选择足够数目的测试数据,使被测程序中每条语句至少执行一次。

2）判定覆盖

判定覆盖又称为分支覆盖或所有边覆盖,基本思想是:设计测试用例,使得程序中每个判定至少分别取“真”分支和取“假”分支各一次,即判断真、假值均被满足。

3）条件覆盖

条件覆盖是指设计测试用例,使每个判断中每个条件的可能取值至少满足一次。

4）判定-条件覆盖

判定-条件覆盖的含义是:通过设计足够的测试用例,使得判断条件中的所有条件可能至少执行一次取值,同时,所有判断的可能结果至少执行一次。

5）条件组合覆盖

条件组合覆盖的基本思想是:设计测试用例,使得判断中每个条件的所有可能组合都至少出现一次。而条件覆盖是简单地要求每个条件都出现“真”与“假”两种结果。

6）路径覆盖

选择足够的测试用例,使得程序中所有的可能路径都至少被执行一次。

逻辑覆盖方法中语句覆盖、判定覆盖、条件覆盖、判定-条件覆盖、条件组合覆盖和路径覆盖具有相互包含的关系,其中语句覆盖最弱,其余依次增强,路径覆盖的效果最好,

如图 4.1 所示。

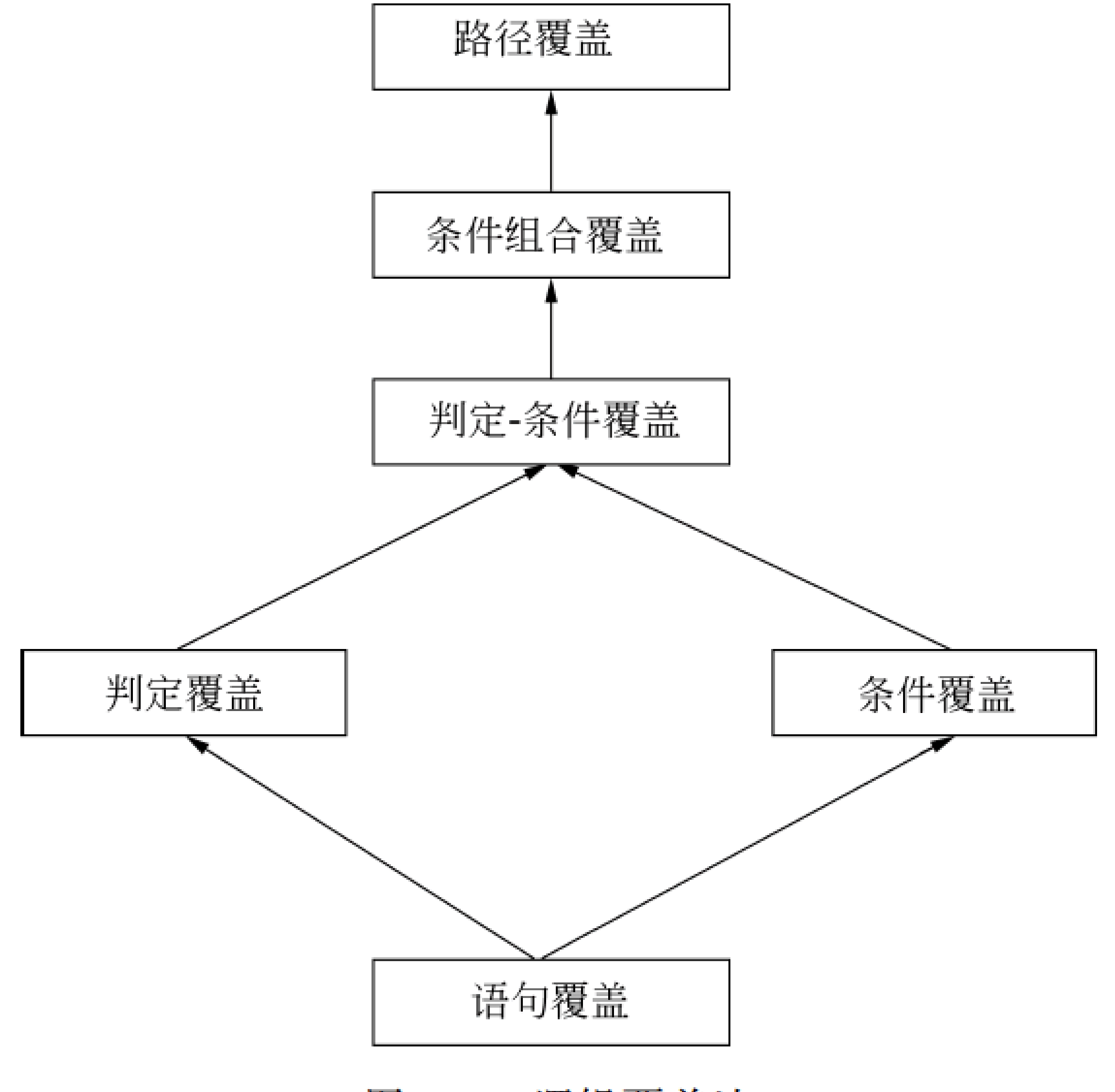


图 4.1 逻辑覆盖法

4. 基本路径测试

基本路径测试使测试用例设计者可以求出程序或过程设计中的逻辑复杂性测度，并使用该测度作为指南来定义执行路径的基本集，从该基本集导出的测试用例保证对程序中的每一条语句至少执行一次。

5. 程序插桩

程序插桩是在被测程序中添加语句，对程序语句中变量值进行检查。程序员往往在程序中插入一些打印语句，通过分析输出信息，了解程序执行过程中的动态特性(例如，程序的实际执行路径或特定变量在特定时刻的取值)，插入的语句往往称为“探测器”。

程序插装技术的研究涉及下列几个问题：

(1) 探测哪些信息?

(2) 在程序的什么位置设置探测点?

(3) 需要多少探测点?

6. 数据流测试

数据流测试主要用于检测程序编写时出现的一些警告信息，如“所定义的变量未被使用等”问题。数据流测试具有如下两种方法：

(1) 变量定义/引用分析。从代码本身的变量定义点与引用点的内在关系出发进行的一种结构性测试，它利用变量之间的关系进行测试。

(2) 程序片。往往用来排除程序的某些无意义的片段，能够较快地定位异常。

4.3 典型习题解析

4.3.1 选择题

1. 以程序内部的逻辑结构为基础的测试用例设计技术属于(　　)。

A. 灰盒测试　　B. 数据测试　　C. 黑盒测试　　D. 白盒测试

【答案】 D

2. 白盒测试是根据程序的(　　)设计测试用例。

A. 功能　　B. 性能　　C. 内部逻辑　　D. 内部数据

【答案】 C

3. 白盒测试方法的优点是(　　)。

A. 可测试软件的特定部位　　B. 能站在用户立场测试

C. 可按软件内部结构测试　　D. 可发现功能需求实现中的错误

【答案】 C

4. 使用白盒测试方法时,测试数据应根据(　　)确定覆盖标准。

A. 程序的内部结构　　B. 程序的复杂性

C. 使用说明书　　D. 程序的功能

【答案】 A

5. 以下针对逻辑覆盖的叙述中(　　)是不正确的。

A. 达到 100% CC 要求就一定能够满足 100% DC 的要求

B. 达到 100% CDC 要求就一定能够满足 100% DC 的要求

C. 达到 100% MCDC 要求就一定能够满足 100% DC 的要求

D. 达到 100%路径覆盖要求就一定能够满足 100% DC 的要求

【答案】 A

6. 以下不属于白盒测试技术的是(　　)。

A. 逻辑覆盖　　B. 基本路径测试　　C. 循环覆盖测试　　D. 等价类划分

【答案】 D

7. 针对程序段

```
IF (X> 10) AND (Y< 20) THEN
        W=W/A
```

对于(X,Y)的取值,以下(　　)测试用例能够满足判定覆盖的要求。

A. (30,15)(40,10)　　B. (3,0)(30,30)

C. (5,25)(10,20)　　D. (20,10)(1,100)

【答案】 D

8. 在用逻辑覆盖法设计测试用例时,有语句覆盖、分支覆盖、条件覆盖、判定-条件覆盖、条件组合覆盖和路径覆盖等。其中(　　)是最强的覆盖准则。

A. 语句覆盖　　B. 条件覆盖　　C. 判定-条件覆盖　　D. 路径覆盖

【答案】 D

9. 以下不属于逻辑覆盖的是(　　)。

A. 语句覆盖　　B. 判定覆盖　　C. 条件覆盖　　D. 基本路径

【答案】 D

10. 条件组合覆盖是一种逻辑覆盖,它的含义是:设计足够的测试用例,使得每个判定中条件的各种可能组合都至少出现一次,满足条件组合的测试用例也是满足(　　)级别的测试。

A. 语句覆盖、判定覆盖、条件覆盖、条件判定组合覆盖

B. 判定覆盖、条件覆盖、条件判定组合覆盖、修正条件判定覆盖

C. 语句覆盖、判定覆盖、条件判定组合覆盖、修正条件判定覆盖

D. 路径覆盖、判定覆盖、条件覆盖、条件判定组合覆盖

【答案】 A

11. 判定覆盖(　　)包含条件覆盖,条件覆盖(　　)包含判定覆盖。

A. 不一定,不一定　　B. 不一定,一定

C. 一定,不一定　　D. 一定,一定

【答案】 A

12. McCabe 建议模块规模应满足 $V(G) \leqslant$(　　)。

A. 20　　B. 10　　C. 30　　D. 40

【答案】 B

13. 下列关于逻辑覆盖的说法中错误的是(　　)。

A. 满足条件覆盖并不一定满足判定覆盖

B. 满足条件组合覆盖的测试一定满足判定覆盖、条件覆盖和判定-条件覆盖

C. 满足路径覆盖也不一定满足条件组合覆盖

D. 满足判定-条件覆盖同时满足判定覆盖和条件覆盖

【答案】 C

14. 对程序的测试最好由(　　)来做,对程序的调试最好由(　　)来做。

A. 程序员,第三方测试机构　　B. 第三方测试机构,程序员

C. 程序开发组,程序员　　D. 程序开发组,程序开发组

【答案】 B

15. 对于(A＞1)or(B＜＝3),为了达到100％的条件覆盖率,至少需要设计(　　)个测试用例。

A. 1　　B. 2　　C. 3　　D. 4

【答案】 B

16. 在覆盖准则中,最常用的是(　　)。

A. 语句覆盖　　B. 条件覆盖　　C. 分支覆盖　　D. 以上全部

【答案】 D

17. 数据流覆盖是(　　)的变种。

A. 语句覆盖　　B. 控制覆盖　　C. 分支覆盖　　D. 路径覆盖

【答案】 D

18. 不属于逻辑覆盖方法的是(　　)。

A. 组合覆盖　　B. 判定覆盖　　C. 条件覆盖　　D. 接口覆盖

【答案】 D

19. (　　)是选择若干个测试用例,运行被测程序,使得程序中的每个可执行语句至少执行一次。

A. 条件覆盖　　B. 组合覆盖　　C. 判定覆盖　　D. 语句覆盖

【答案】 D

20. (　　)是设计足够多的测试用例,使得程序中每个判定包含的每个条件的所有情况(真、假)至少出现一次,并且每个判定本身的判定结果(真、假)也至少出现一次。

A. 判定-条件覆盖　　B. 组合覆盖

C. 判定覆盖　　D. 条件覆盖

【答案】 A

21. (　　)是指为查明程序中的错误和缺陷可能使用的工具和手段。

A. 调试技术　　B. 测试技术　　C. 跟踪法　　D. 动态测试

【答案】 A

22. 从已发现故障的存在,到找到准确的故障位置,并确定故障的性质,这一过程称为(　　)。

A. 错误检测　　B. 故障排除　　C. 调试　　D. 测试

【答案】 C

23. 下列关于测试与调试的说法中错误的是(　　)。

A. 测试是显示错误的行为,而调试是推理的过程

B. 测试显示开发人员的错误,调试是开发人员为自己辩护

C. 测试能预期和可控,调试需要想象、经验和思考

D. 测试必须在详细设计已经完成的情况下才能开始,没有详细设计的信息不可能进行调试

【答案】 D

4.3.2　简答题

1. 白盒测试是什么?

【答】 白盒测试方法也称结构测试或逻辑驱动测试。白盒测试方法是根据对模块内部结构的了解,基于内部逻辑结构,针对程序语句、路径、变量状态等进行测试,检验程序中的各个分支条件是否得到满足,每条执行路径是否按预定要求正确地工作。

2. 如果能够执行完美的黑盒测试,还需要进行白盒测试吗? 为什么?

【答】 首先人不是机器,不可能进行完美的黑盒测试,更何况机器也有出错的时候。黑盒测试主要是对软件的功能和性能方面的测试,覆盖测试其全部路径,而白盒测试可以

发现软件的内部结构问题，这是黑盒测试所做不到的，就其覆盖路径测试方面，白盒测试也比黑盒测试执行的效率要高。从软件的生命周期来看，进行白盒测试能缩短软件开发时间，节约开发费用。

3. 为什么说语句覆盖是最弱的逻辑覆盖？

【答】 语句覆盖测试方法仅仅针对程序逻辑中的显式语句，对隐藏条件无法测试。例如，逻辑运算符 and 误写成 or，设计测试用例虽仍能达到语句覆盖的要求，但是并未发现程序中的拼写错误。另外，该方法对一些控制结构不敏感，不能发现判断中逻辑运算符出现的错误。

4. 条件覆盖为什么不一定包含判定覆盖？

【答】 条件覆盖只能保证每个条件至少有一次为真，而不考虑所有的判定结果。满足条件覆盖的测试用例由于所有判定结果都是 False，并没有满足判定覆盖。所以条件覆盖不一定包含判定覆盖。

5. 条件判定覆盖(C/DC)与修正条件判定覆盖(MC/DC)有哪些差异？

【答】 C/DC 方法只要求"判定中每个条件的所有可能取值至少执行一次，同时每个判定的所有可能判定结果至少执行一次"，要求比较简单。而 MC/DC 方法是在 C/DC 方法上的改进，它的覆盖面大于 C/DC 方法，也就是在测试一个程序或一个软件时，MC/DC 能比 C/DC 找到更多的错误之处。由于 MC/DC 在寻找测试集时的要求比 C/DC 严格，所以所花费的时间多，MC/DC 适合那些大型的并且要求测试非常精确的软件测试，主要应用于大型的航空航天软件程序的测试上。C/DC 方法要求较低，开销少，而覆盖率也低。

6. 数据流测试是什么？

【答】 数据流测试从代码的内在关系出发进行结构性测试，它是利用变量之间的关系进行测试。作为路径覆盖的变异，考虑从变量定义到变量引用之间的路径。

7. 数据流测试的定义/引用方法与程序片有什么不同？

【答】 数据流测试的定义/引用方法基于路径，应用于计算密集的程序，具有结构化的全局特性，故能反映出程序代码的全局异常。程序片往往只能反应局部状况。

8. 白盒测试中为什么不适用穷举测试？

【答】 第一，穷举路径测试不能查出程序违反了设计规范，即程序本身是一个错误的程序。第二，穷举路径测试不可能查出程序中因遗漏路径而出错的情况。第三，穷举路径测试可能发现不了一些与数据相关的错误。

9. 软件调试执行的步骤是什么？

【答】

（1）从错误的外部表现形式入手，确定程序中出错位置。

（2）研究有关部分的程序，找出错误的内在原因。

（3）修改设计和代码，以排除这个错误。

（4）重复暴露这个错误的原始测试或某些有关测试，以确认是否排除了该错误或是否引进了新的错误。

（5）如果所做的修正无效，则撤销这次改动，恢复程序修改之前的状态。

10. 测试与调试的关系是什么？

【答】 测试的目的是显示存在错误，而调试的目的是发现错误或导致程序失效的原因，并修改程序以修正错误。调试是测试之后的活动。测试和调试在目标、方法和思路上都有所不同，具体如下：

（1）测试过程可以事先设计，进度可事先确定；调试不能描述过程或持续时间。

（2）测试是显示错误的行为，调试是推理的过程。

（3）测试显示开发人员的错误，调试是开发人员为自己辩护。

（4）测试能预期和可控，调试需要想象、经验和思考。

（5）测试能在没有详细设计的情况下完成，没有详细设计的信息不可能进行调试。

（6）测试能由非开发人员进行，调试必须由开发人员进行。

11. 白盒测试中测试方法的选择有哪些策略？

【答】

（1）在测试中，首先尽量使用测试工作进行静态结构分析。

（2）采用先静态后动态方式。先进行静态结构分析、代码检查和静态质量度量，然后进行覆盖测试。

（3）利用静态结构分析的结果，通过代码检查和动态测试的方法对结果进一步确认。

（4）使用基本路径测试达到语句覆盖标准；对于重点模块，应使用多种覆盖标准。

（5）不同测试阶段的侧重点不同。

12. 程序插桩是什么？

【答】 程序插桩是在被测程序中添加语句，对程序语句中的变量值进行检查。程序员往往在程序中插入一些打印语句，通过分析输出信息，了解程序执行过程中的动态特性（例如，程序的实际执行路径或特定变量在特定时刻的取值），插入的语句往往称为“探测器”。

4.3.3 设计题

1. 请把图 4.2 所示的程序流程图转化成控制流图。

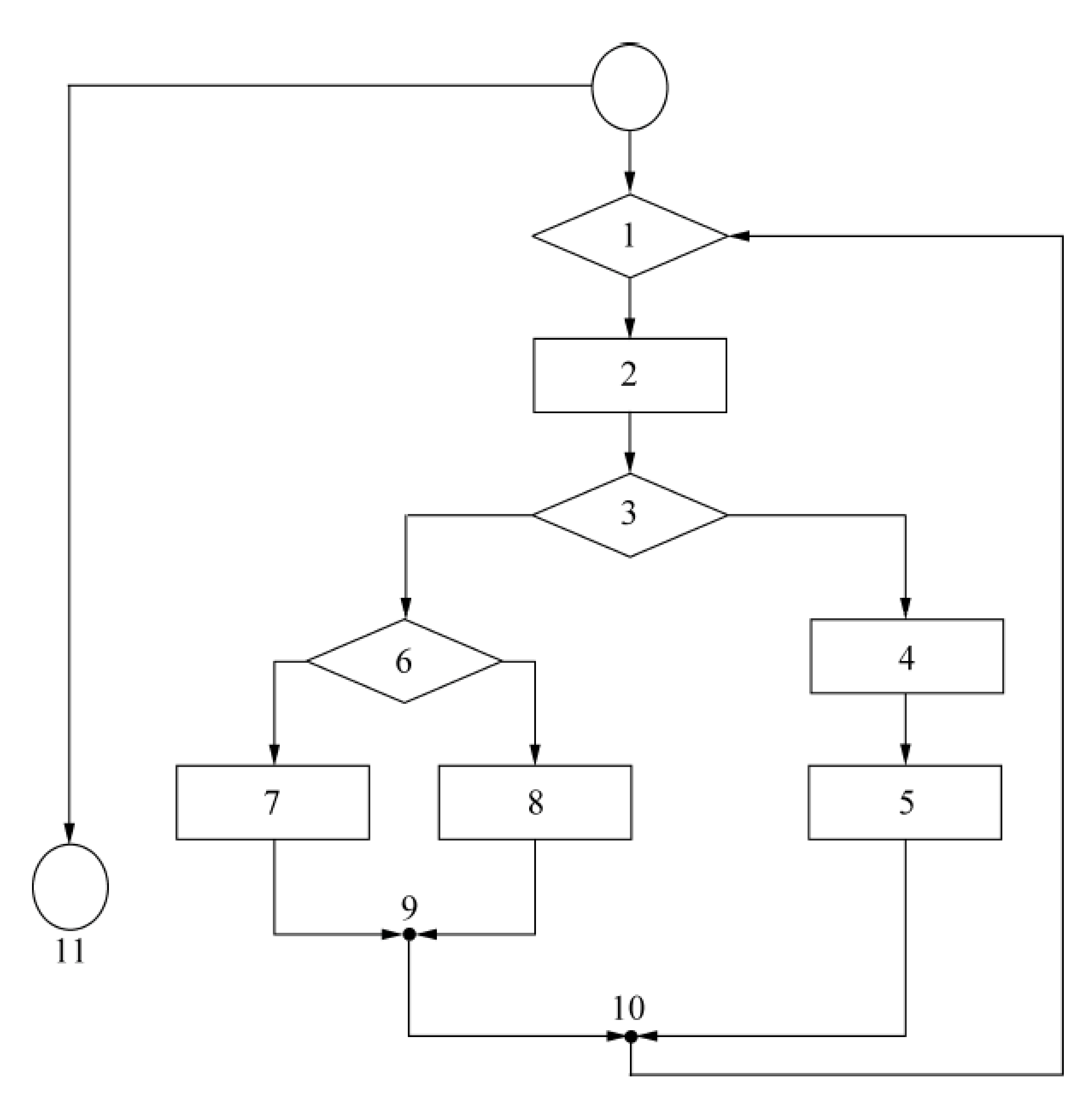


图 4.2 程序流程图

【解析】 程序流程图转化为控制流图,如图 4.3 所示。

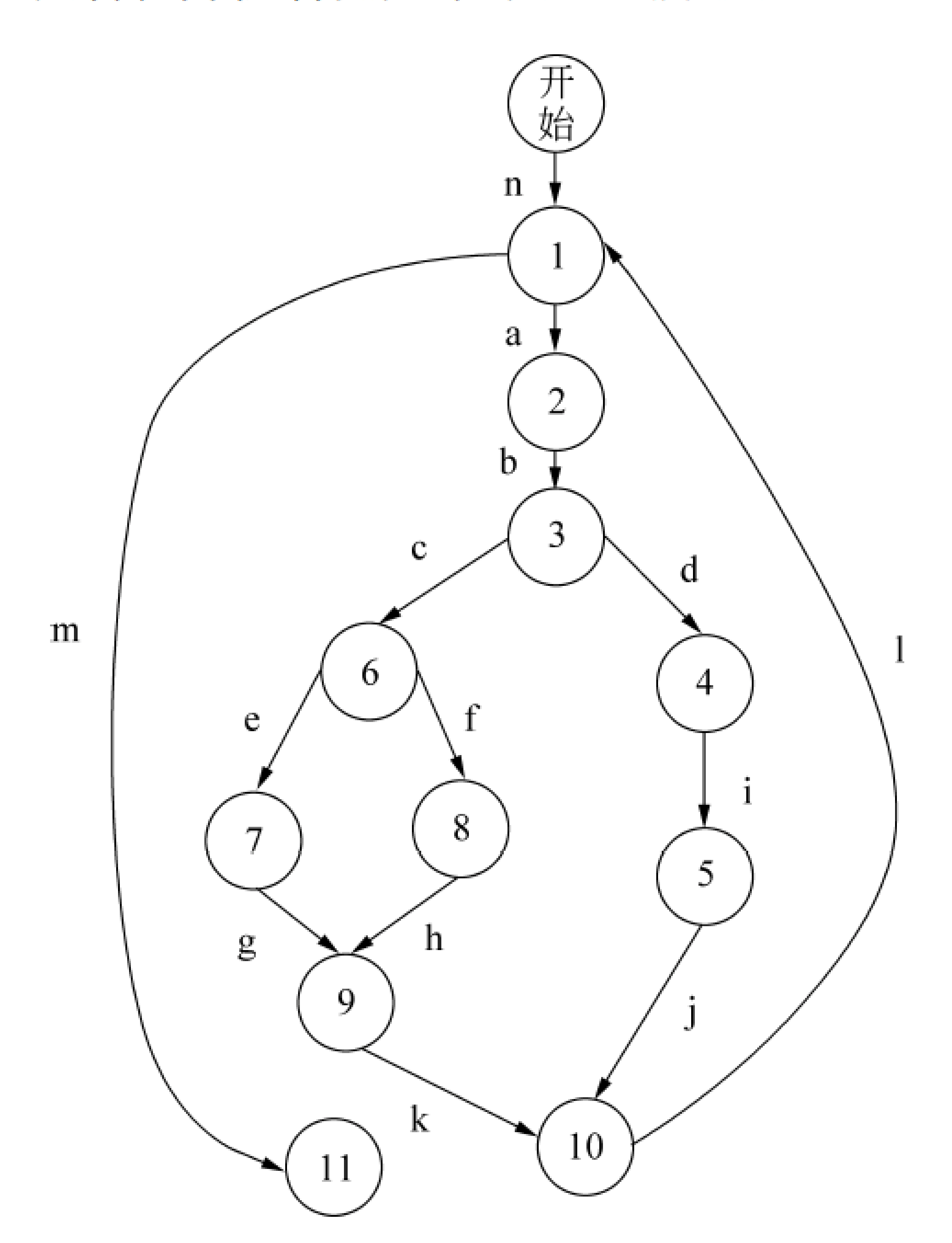


图 4.3 控制流图

2. 有如下子程序:

```
procedure example(y,z: real; var x: real)
begin
  if(y>1) and (z=0)then x:=x/y;
  if(y=2) or (x=1)then x:=x+1;
end
```

要求：

（1）画出程序流程图。

（2）用白盒测试法中的条件组合覆盖设计测试用例。

【解析】

（1）程序流程图如图 4.4 所示。

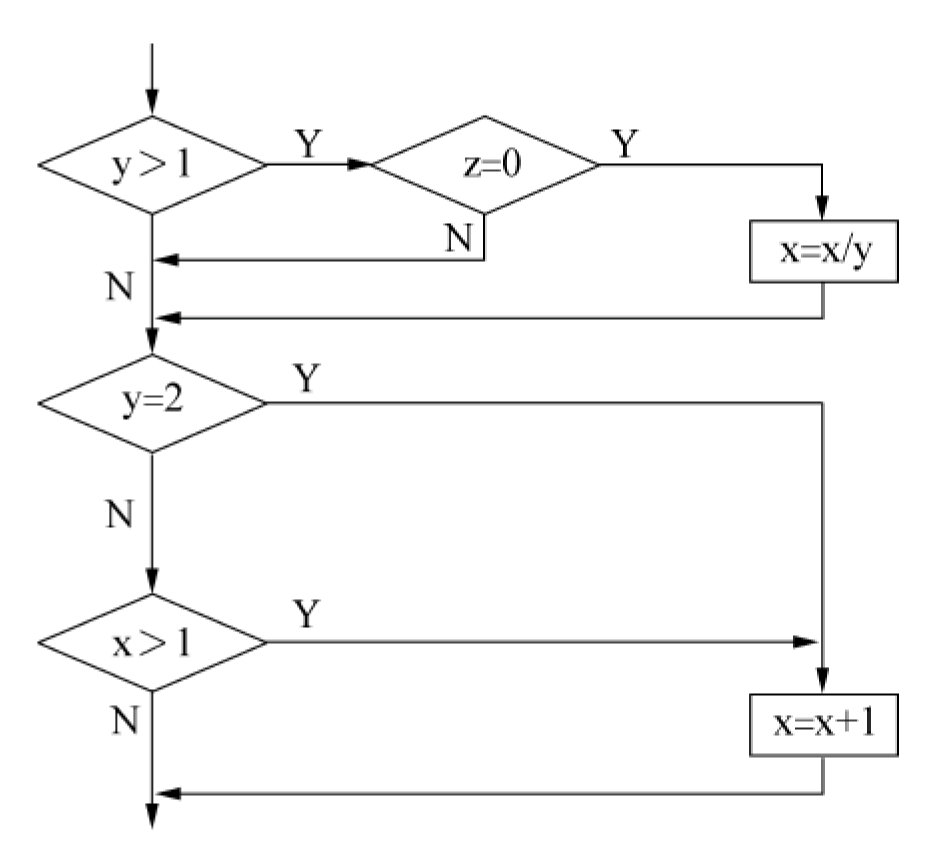


图 4.4　程序流程图

（2）用条件组合覆盖设计测试用例：

① y=2，z=0，x=4。

② y=2，z=1，x=1。

③ y=1，z=0，x=2。

④ y=1，z=1，x=1。

3. 控制流图如图 4.5 所示，请给出其所有的独立路径。

【解析】 采用红色、绿色、蓝色和黑色 4 种颜色绘制出 4 条独立路径，如图 4.6 所示。

独立路径 1：1-11。

独立路径 2：1-2-3-4-5-10-1-11。

独立路径 3：1-2-3-6-8-9-10-1-11。

独立路径 4：1-2-3-6-7-9-10-1-11。

4. 有如下程序段：

```
void ReadPara(CString temp)
{
  if(temp==">=")
    m_oper.SetCurSel(0);
```

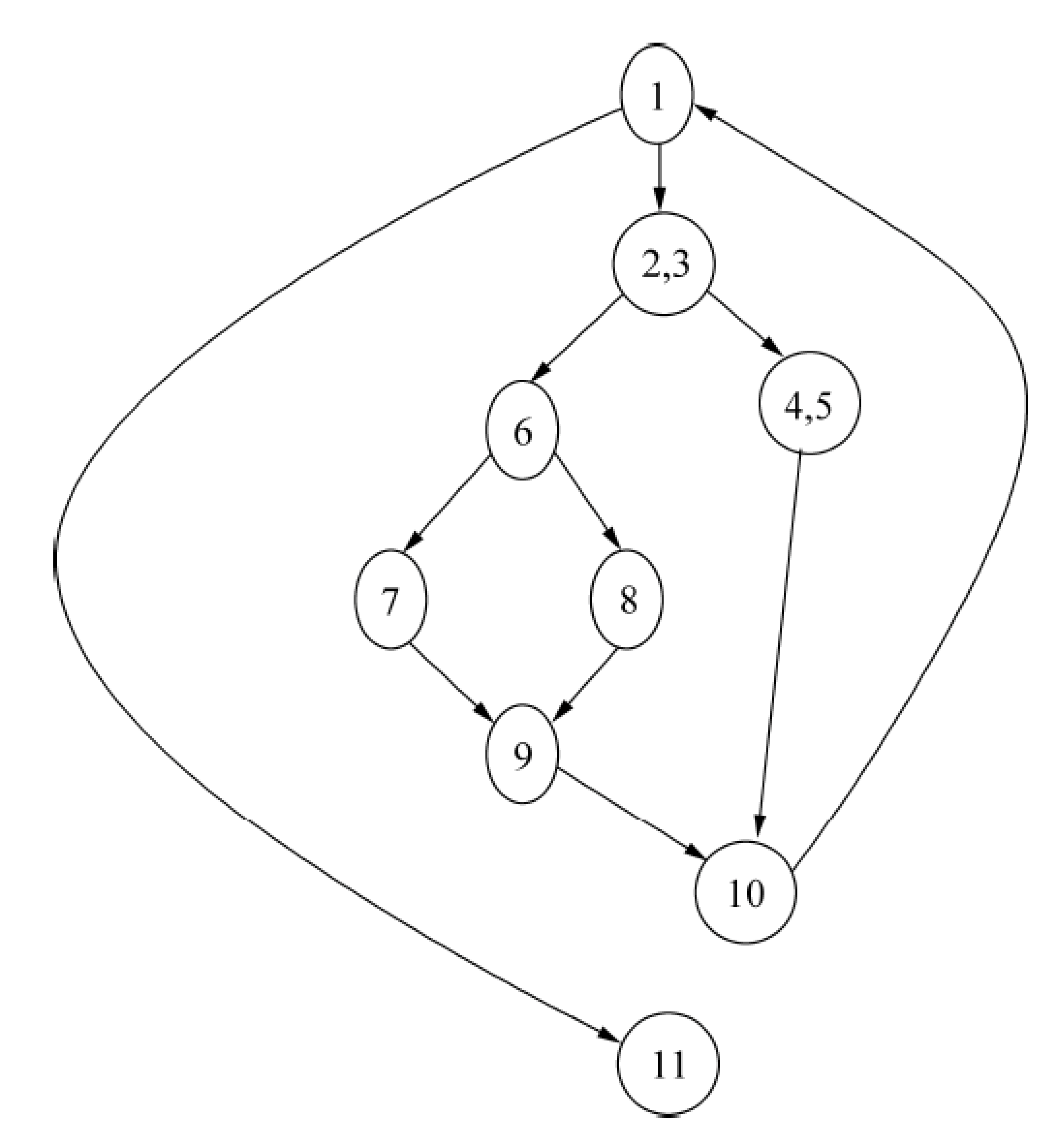


图 4.5 控制流图

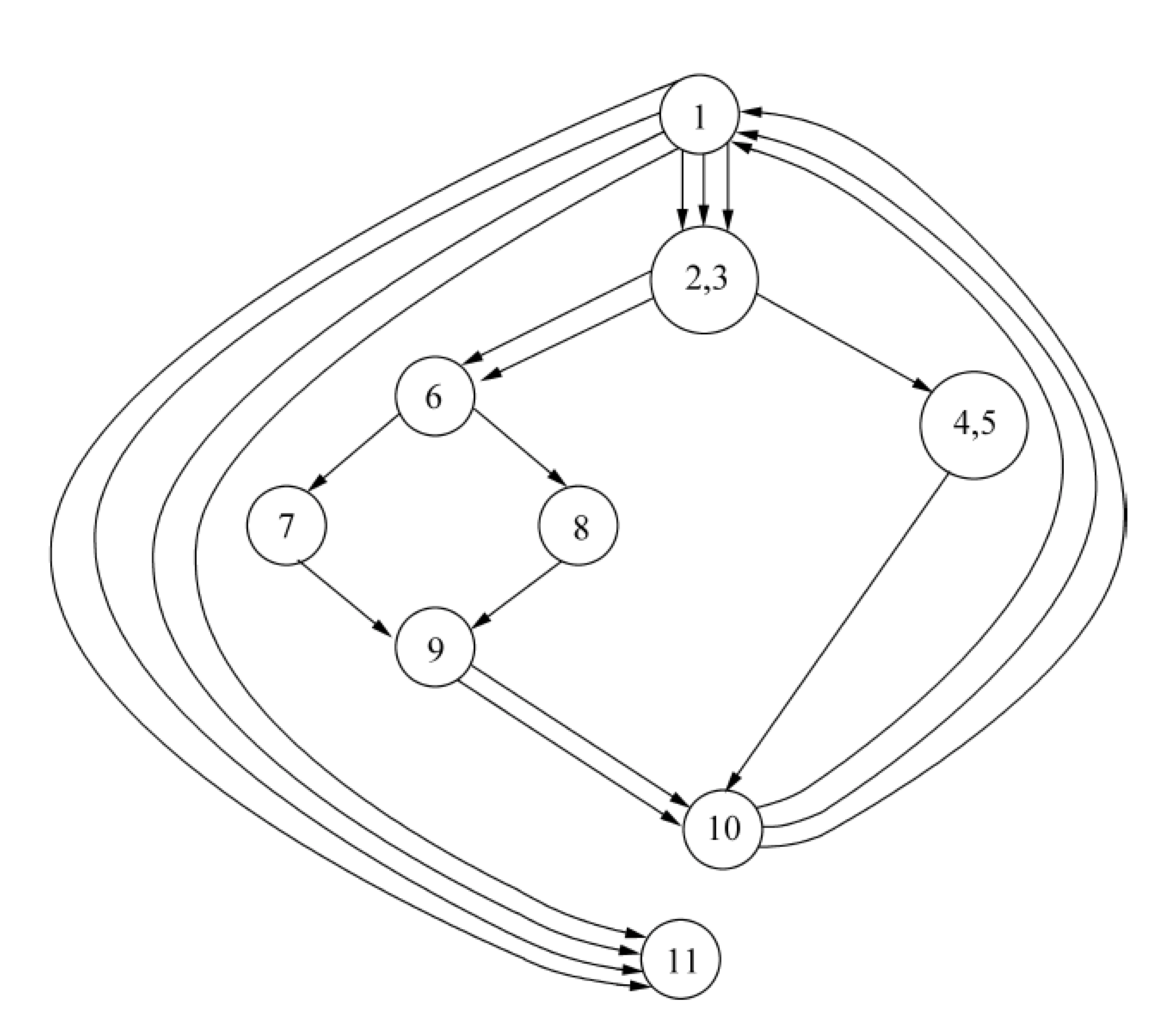


图 4.6 控制流图

```
else
{
  if(temp==">")
    m_oper.SetCurSel(1);
  else
  {
    if(temp=="==")
      m_oper.SetCurSel(2);
```

```
            else
            {
               if(temp=="<=")
                    m_oper.SetCurSel(3);
               else
               {
                    if(temp=="<")
                        m_oper.SetCurSel(4);
                    else
                        m_oper.SetCurSel(5);
               }
            }
         }
      }
      return;
  }
```

要求：

（1）画出这段代码的控制流图。

（2）计算环路复杂度 $V(G)$。

（3）列出独立路径。

（4）设计测试用例。

【解析】

（1）这段代码的控制流图如图 4.7 所示。

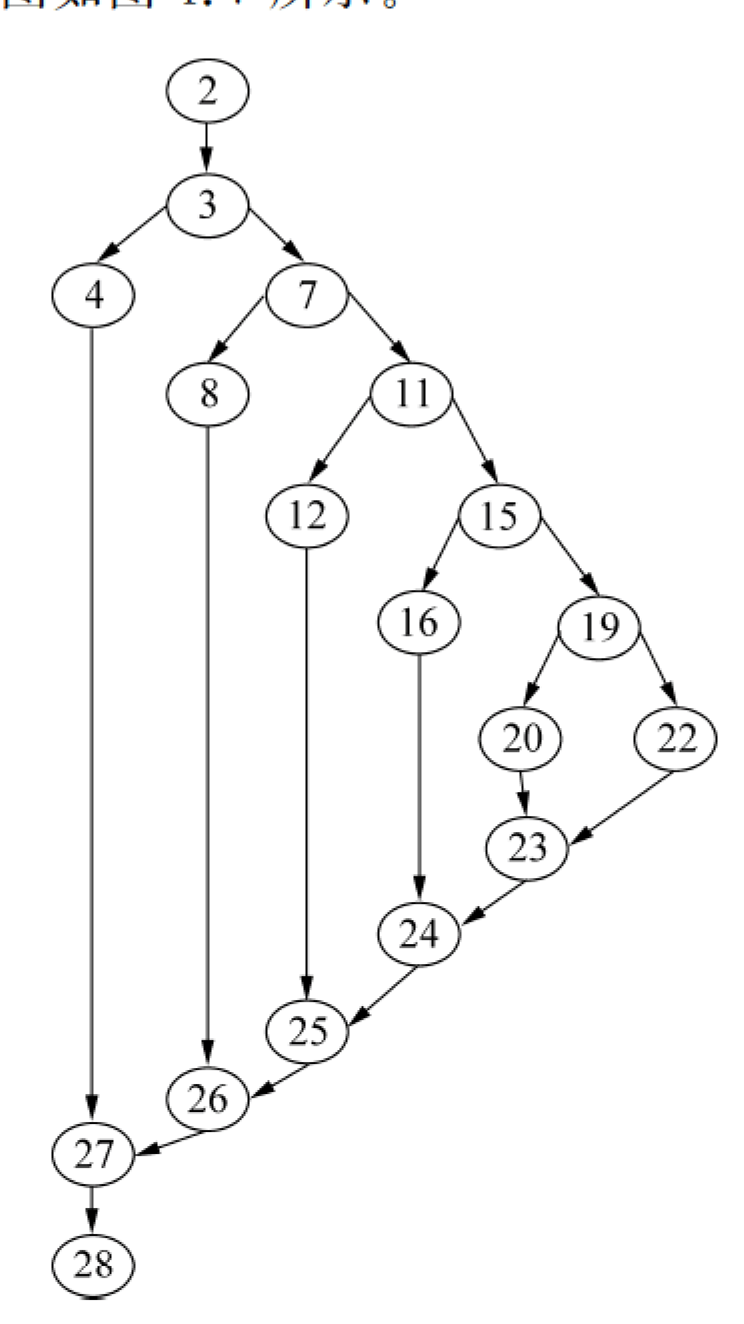


图 4.7　控制流图

(2) 环路复杂度 $V(G)=E-N+2=22-18+2=6$。

(3) 独立路径如下:

Path1: 2-3-4-27-28。

Path2: 2-3-7-8-26-27-28。

Path3: 2-3-7-11-12-25-26-27-28。

Path4: 2-3-7-11-15-16-24-25-26-27-28。

Path5: 2-3-7-11-15-19-20-23-24-25-26-27-28。

Path6: 2-3-7-11-15-19-22-23-24-25-26-27-28。

(4) 根据第(3)步中给出的路径,设计测试用例,如表 4.2 所示。

表 4.2 测试用例

路　径	传入参数	预期调用
Path 1	ReadPara(">=")	m_oper. SetCurSel(0)
Path 2	ReadPara(">")	m_oper. SetCurSel(1)
Path 3	ReadPara("==")	m_oper. SetCurSel(2)
Path 4	ReadPara("<")	m_oper. SetCurSel(3)
Path 5	ReadPara("<=")	m_oper. SetCurSel(4)
Path 6	ReadPara("+")	m_oper. SetCurSel(5)

5. 程序流程图如图 4.8 所示,请计算其独立路径。

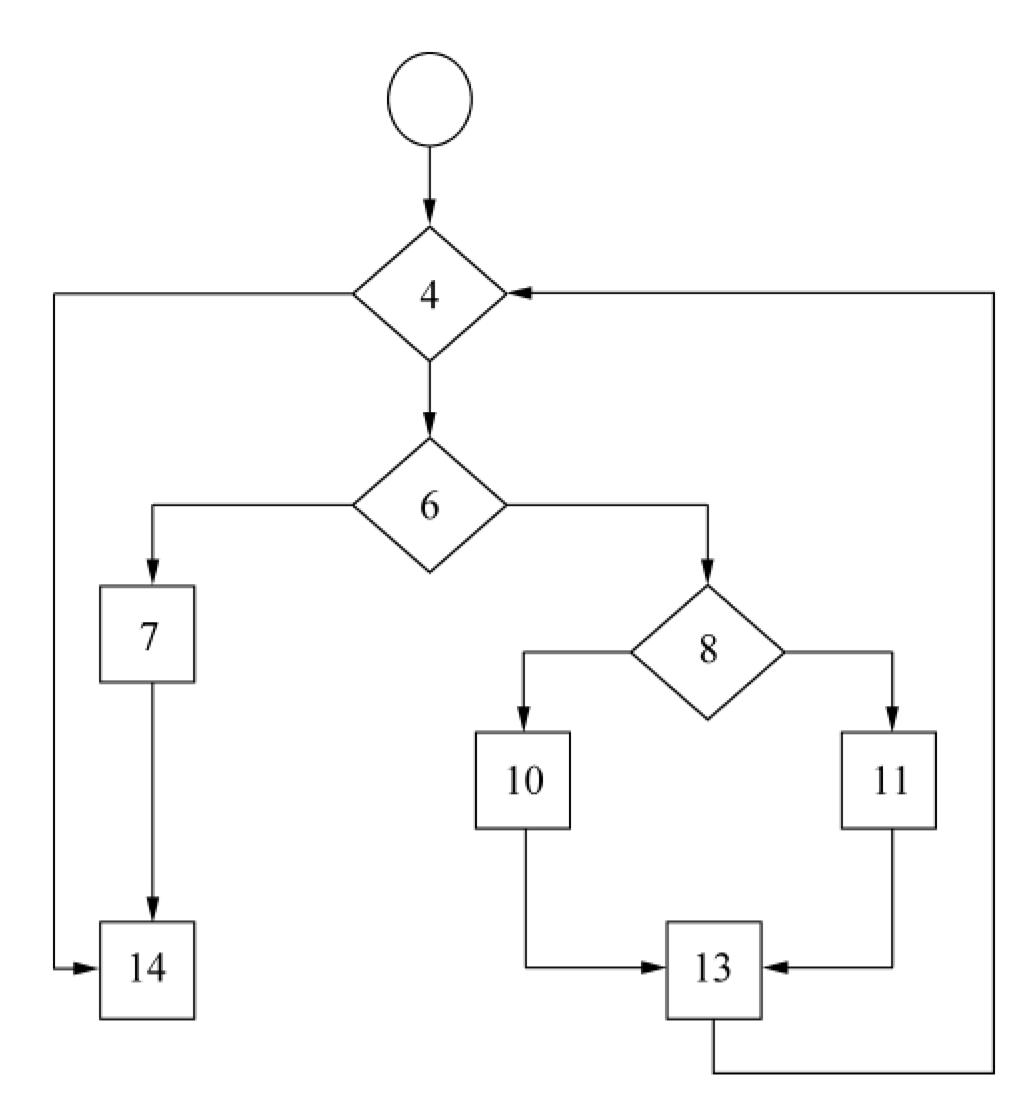


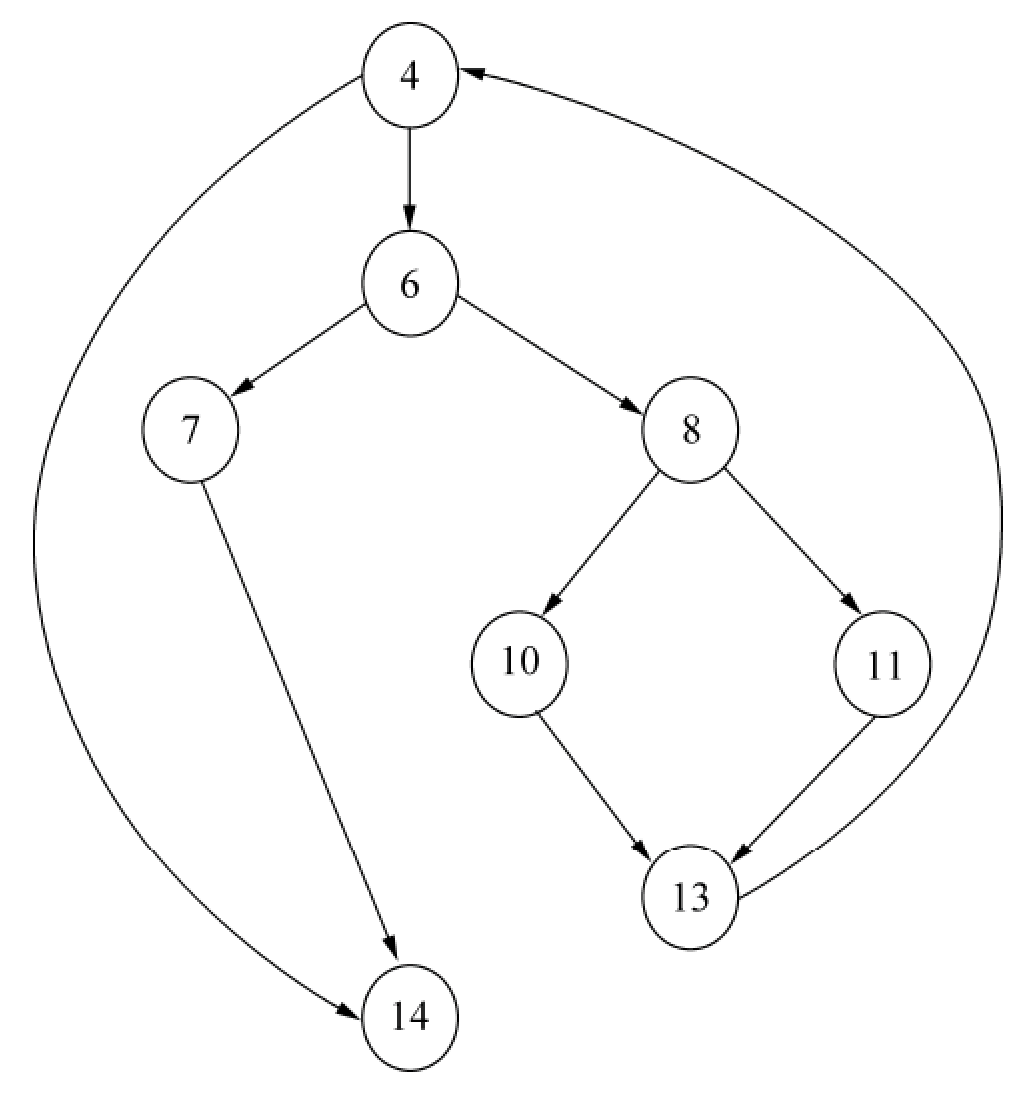
图 4.8 程序流程图

【解析】 基本路径测试法的步骤如下。

步骤 1: 画出控制流图。

程序流程图中的处理方框序列和菱形决策框可以映射为一个圆,在控制流图中称为

结点。控制流图中的箭头称为边或连接,代表控制流,类似于程序流程图中的箭头。控制流图如图 4.9 所示。



4.9 控制流图

步骤 2:计算圈的复杂度。

圈的复杂度用于计算程序基本的独立路径数目,它是确保所有语句至少执行一次的测试数量的上界。独立路径必须包含一条在定义之前不曾用到的边。

用如下方法计算流图 G 的圈复杂度 $V(G)$,如图 4.10 所示。

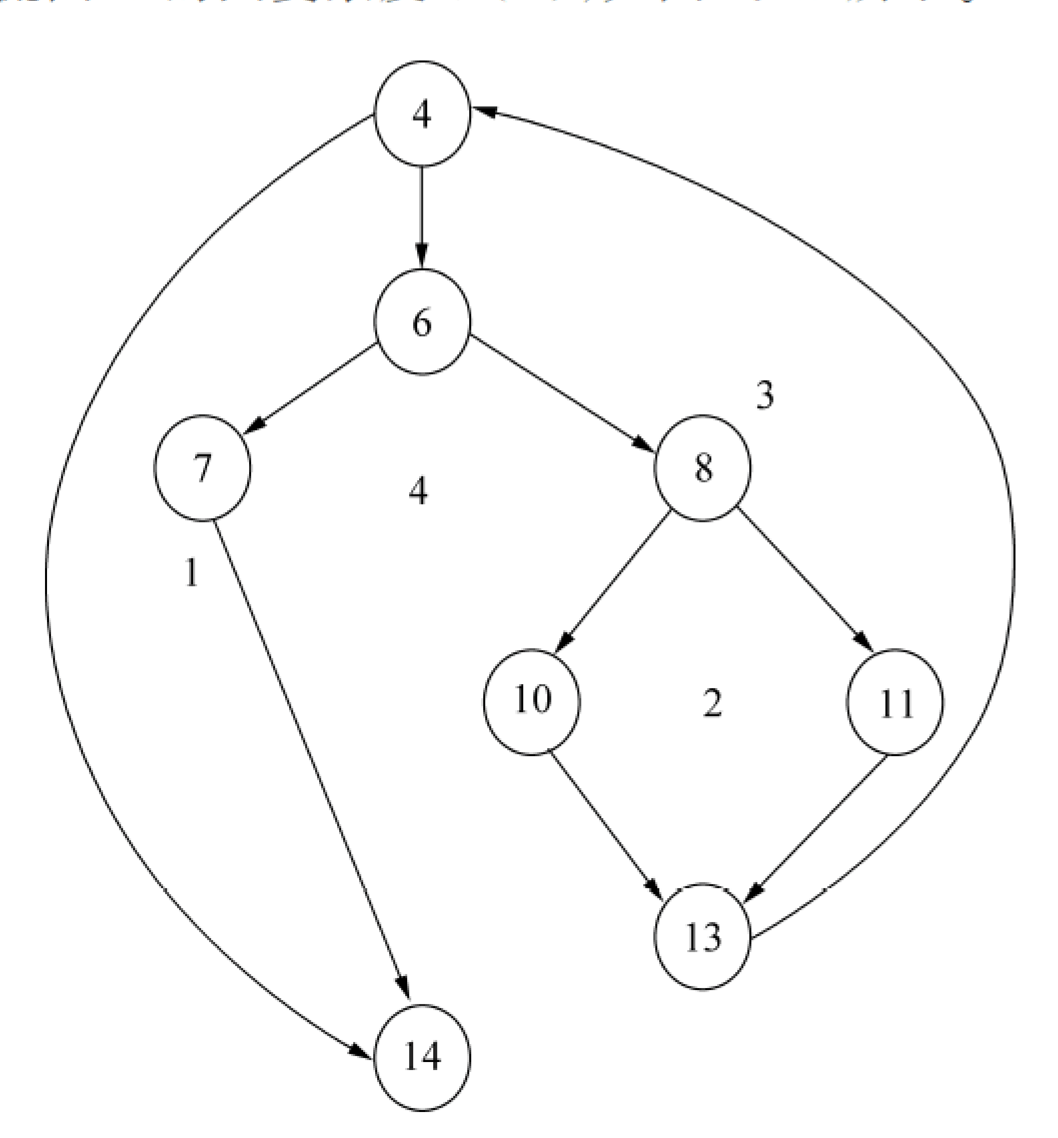


图 4.10 控制流图

(1) 流图中区域的数量对应于圈的复杂度。

(2) 定义 $V(G)=E-N+2$,E 是流图中边的数量,N 是流图中结点的数量。

(3) 定义 $V(G)=P+1$,P 是流图 G 中判定结点的数量。

对应图 4.10 中的圈的复杂度,计算如下:

(1) 控制流图中有 4 个区域;

(2) $V(G)=E-N+2=10-8+2=4$。

(3) $V(G)=P+1=3+1=4$。

步骤 3:计算独立路径。

根据圈的复杂度得出 4 个独立的路径。

路径 1:4-14。

路径 2:4-6-7-14。

路径 3:4-6-8-10-13-4-14。

路径 4:4-6-8-11-13-4-14。

6. 某程序的逻辑结构如图 4.11 所示。设计足够的测试用例实现对程序的判定覆盖、条件覆盖和条件组合覆盖。

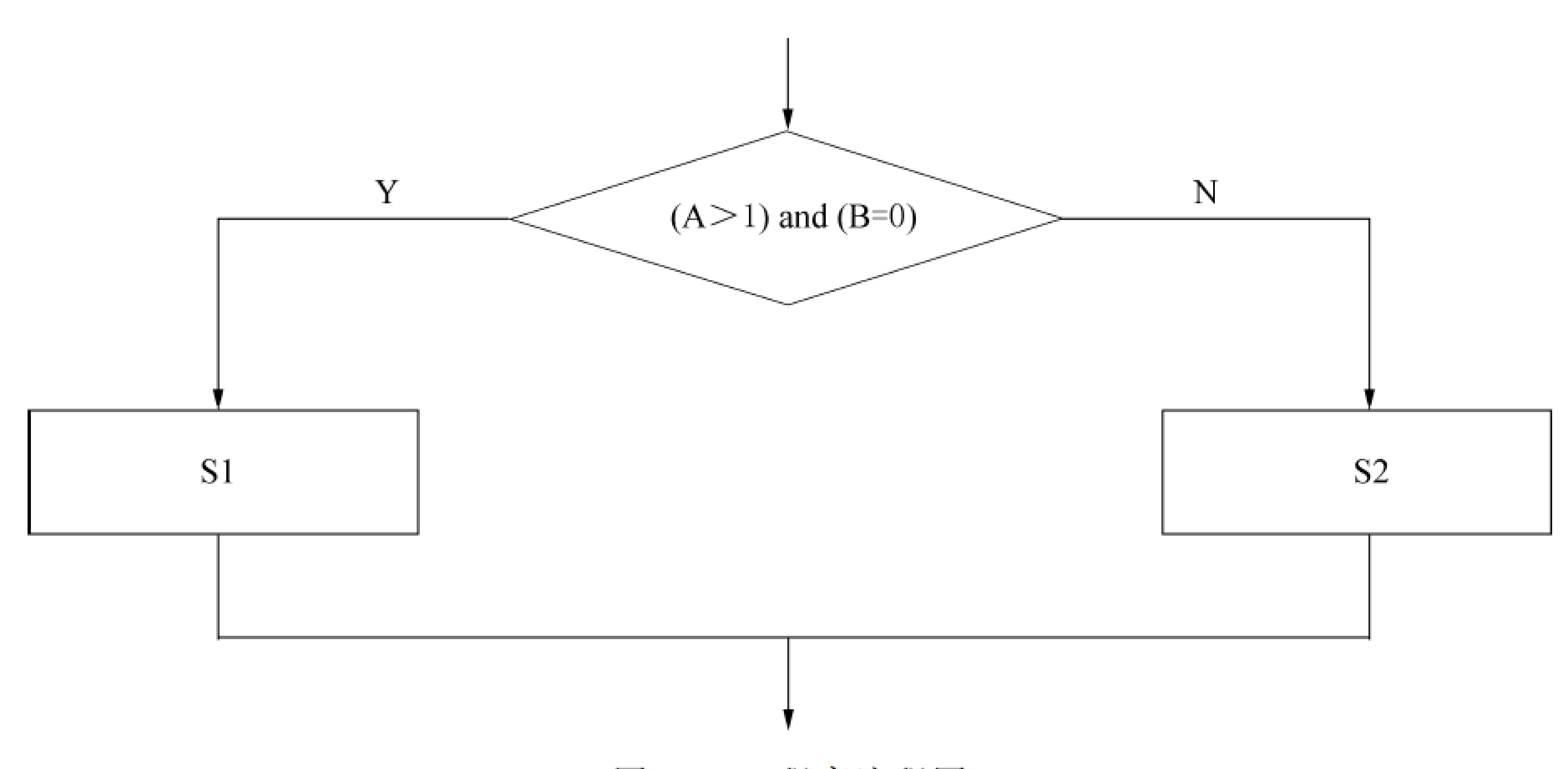


图 4.11 程序流程图

【解析】 设计测试用例,如表 4.3 所示。

表 4.3 逻辑覆盖测试用例

覆盖种类	需满足的条件		测试数据	期望结果
判定覆盖	A>1,B=0		A=2,B=0	执行 S1
	A>1,B≠0 或 A≤1,B=0 或 A≤1,B≠0		A=2,B=1 或 A=1,B=0 或 A=1,B=1	执行 S2
条件覆盖	以下四种情况各出现一次		无	
	A>1	B=0	A=2,B=0	执行 S1
	A≤1	B≠0	A=1,B=1	执行 S2

续表

覆盖种类	需满足的条件	测试数据	期望结果
条件组合覆盖	A>1,B=0	A=2,B=0	执行 S1
	A>1,B≠0	A=2,B=1	执行 S2
	A≤1,B=0	A=1,B=0	执行 S2
	A≤1,B≠0	A=1,B=1	执行 S2

7. 使用逻辑覆盖法测试如下程序段：

```
void work(int x,int y,int z){
  int k=0,j=0;
  if((x>3)&&(z<10)){
     k=x*y-1;
     j=k-z;
  }
  if((x==4)||(y>5)){
     j=x*y+10;
  }
  j=j%3;
}
```

【解答】

步骤 1：将源代码转化为程序流程图，如图 4.12 所示。

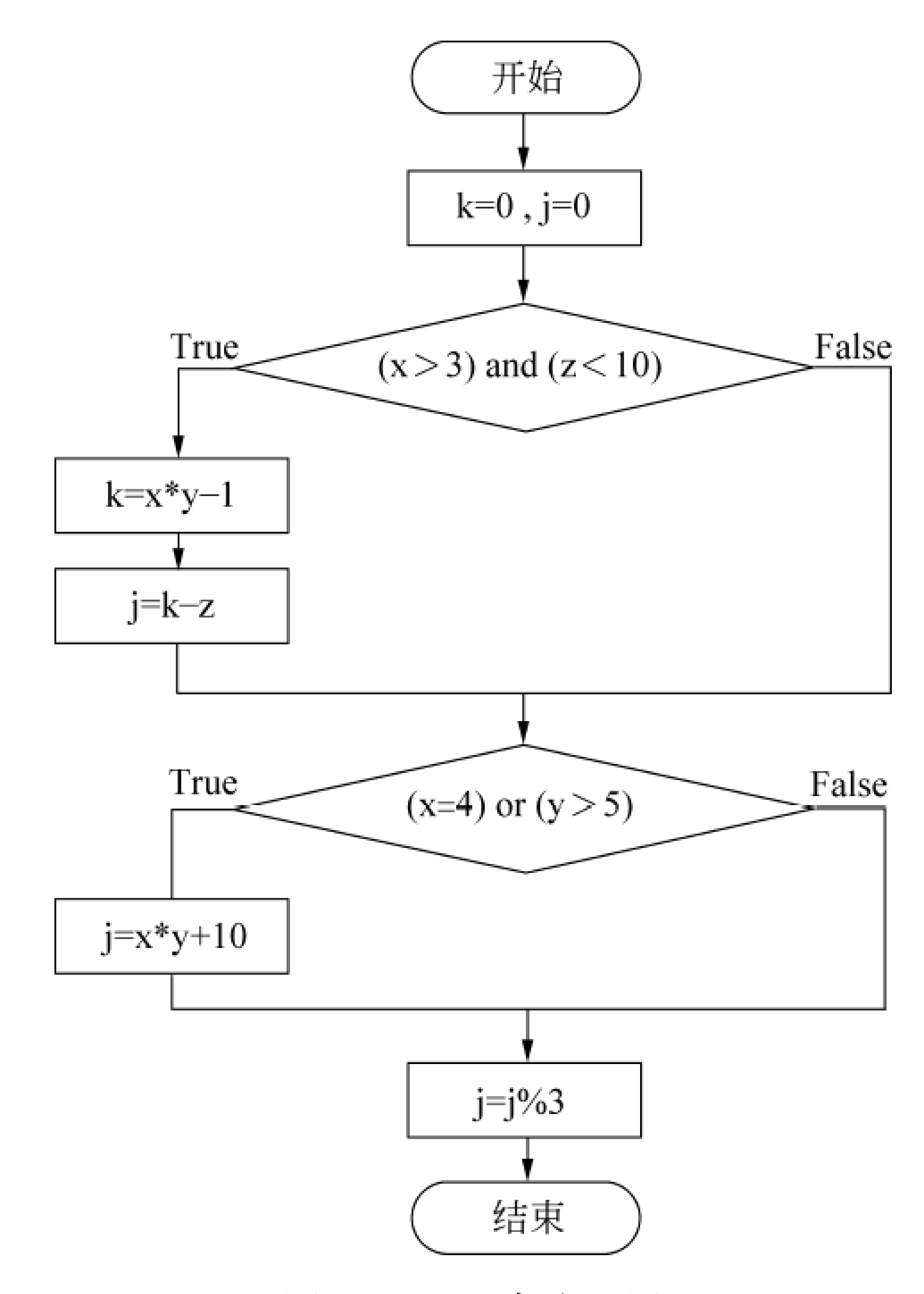


图 4.12　程序流程图

步骤2：将程序流程图转化为控制流图，如图4.13所示。

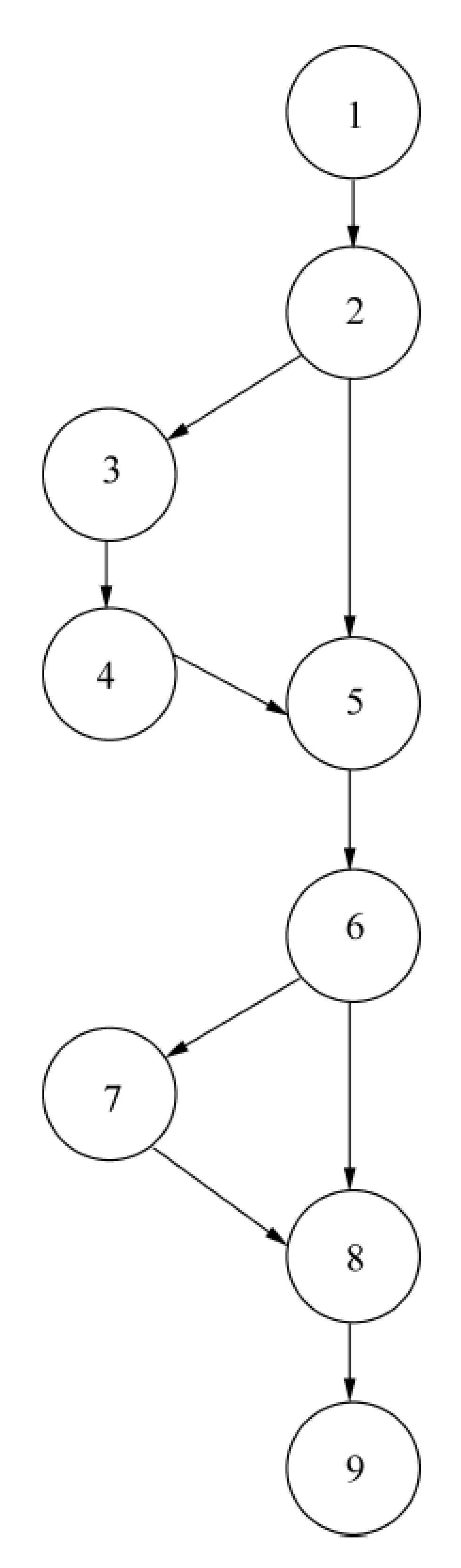


图4.13 控制流图

步骤3：按照不同的覆盖准则设计测试用例。

(1) 语句覆盖。设计测试用例如下：

x=4,y=5,z=5

(2) 判定覆盖。设计测试用例如下：

x=4,y=5,z=5,其执行路径为：1—2—3—4—5—6—7—8。

x=2,y=5,z=5,其执行路径为：1—2—5—6—8。

(3) 条件覆盖。设计测试用例如下：

① 对于第一个判定((x>3)&&(z<10))：

- 条件X>3取真值记为T1,取假值记为-T1。
- 条件z<10取真值记为T2,取假值记为-T2。

② 对于第一个判定((x=4)||(y>5))：

条件X==4取真值记为T3,取假值记为-T3。

条件y>5取真值记为T4,取假值记为-T4。

条件覆盖要使上述4个条件可能产生的8种情况至少满足一次,设计测试用例,如表4.4所示。

表 4.4 条件覆盖测试用例

测试用例	执行路径	覆盖条件
x=4,y=6,z=5	1—2—3—4—5—6—7—8	T1、T2、T3、T4
x=2,y=5,z=15	1—2—5—6—8	－T1、－T2、－T3、－T4

(4) 判定-条件覆盖。设计测试用例如下:

x=4,y=6,z=5,其执行路径为:1—2—3—4—5—6—7—8。

x=2,y=5,z=15,其执行路径为:1—2—5—6—8,

(5) 条件组合覆盖。各个判定的条件取值组合标记如下:

① x>3,z<10 记为 T1,T2,第一判定取值为真。

② x>3,z>=10 记为 T1,－T2,第一判定取值为假。

③ x<=3,z<10 记为－T1,T2,第一判定取值为假。

④ x<=3,z>=10 记为－T1,－T2,第一判定取值为假。

⑤ x==4,y>5 记为 T3,T4,第二判定取值为真。

⑥ x==4,y<=5 记为 T3,－T4,第二判定取值为真。

⑦ x! =4,y>5 记为－T3,T4,第二判定取值为真。

⑧ x! =4,y<=5 记为 T3,T4,第二判定取值为假。

根据条件组织覆盖的基本思想设计测试用例,如表 4.5 所示。

表 4.5 条件组合覆盖测试用例

测试用例	执行路径	覆盖条件
x=4,y=6,z=5	1—2—3—4—5—6—7—8	T1、T2、T3、T4
x=4,y=5,z=15	1—2—5—6—7—8	T1、－T2、T3、－T4
x=2,y=6,z=5	1—2—5—6—7—8	－T1、T2、－T3、T4
x=2,y=5,z=15	1—2—5—6—8	－T1、－T2、－T3、－T4

(6) 路径覆盖。根据路径覆盖的基本思想设计测试用例,如表 4.6 所示。

表 4.6 路径覆盖测试用例

测试用例	执行路径	覆盖条件
x=4,y=6,z=5	1—2—3—4—5—6—7—8	T1、T2、T3、T4
x=4,y=5,z=15	1—2—5—6—7—8	T1、－T2、T3、－T4
x=5,y=5,z=5	1—2—3—4—5—6—8	T1、T2、－T3、－T4
x=2,y=5,z=15	1—2—5—6—8	－T1、－T2、－T3、－T4



软件测试流程

5.1 本 章 要 求

- 了解软件测试生命周期
- 了解软件测试执行过程
- 掌握单元测试
- 理解集成测试、系统测试、验收测试
- 了解评估测试

5.2 本章知识重点

软件测试生命周期具体包括以下 4 个阶段：

(1) 测试计划。

根据用户需求报告中关于功能要求和性能指标的规格说明书，定义相应的测试需求报告，使得随后所有的测试工作都围绕测试需求进行。同时，适当选择测试内容，合理安排测试人员、测试时间及测试资源等。

(2) 测试设计。

测试设计是指将测试计划阶段制定的测试需求分解、细化为若干个可执行的测试过程，并为每个测试过程选择适当的测试用例，保证测试结果的有效性。

(3) 测试执行。

测试执行阶段建立自动测试过程，并对所发现的缺陷进行跟踪管理。测试执行一般由单元测试、集成测试、系统测试、验收测试以及回归测试等步骤组成。

① 单元测试

目的：检测程序模块中有无故障存在。

对象：软件设计的最小单位，与程序设计和编程实现关系密切。

② 集成测试。

目的：发现与接口有关的模块之间的问题。

方法：非增式集成测试法和增式集成测试法。

③ 系统测试。

目的：针对系统中各个组成部分进行的综合性检验，证明系统的性能。

④ 验收测试。

目的：向用户表明所开发的软件系统能够像用户所预期的那样工作

主要任务：

- 明确规定验收测试通过的标准。
- 确定验收测试方法。
- 确定验收测试的组织和可利用的资源。
- 确定测试结果的分析方法。
- 制定验收测试计划并进行评审。
- 设计验收测试的测试用例。
- 审查验收测试的准备工作。
- 执行验收测试。
- 分析测试结果，确定是否通过验收。

（4）测试评估。

结合量化的测试覆盖域及缺陷跟踪报告，对于应用软件的质量、开发团队的工作进度及工作效率进行综合评价。

5.3 典型习题解析

5.3.1 选择题

1. 软件测试是软件质量保证的重要手段，（　　）是软件测试最基础的环节。

A. 功能测试　　B. 单元测试　　C. 结构测试　　D. 确认测试

【答案】 B

2. 测试计划的制订必须注重（　　）

A. 测试策略、测试范围　　B. 测试方法、测试安排

C. 测试风险、测试治理　　D. 以上都对

【答案】 D

3. 软件测试计划的内容应包括（　　）。

A. 测试目的、背景　　B. 被测软件的功能、输入和输出

C. 测试内容和评价标准　　D. 以上全对

【答案】 D

4. 软件测试计划描述了（　　）。

A. 软件的性质

B. 软件的功能和测试用例

C. 软件的规定动作

D. 对于预定的测试活动将要采取的手段

【答案】 D

5. 软件设计阶段的测试主要采取的方式是（　　）。

A. 评审　　B. 白盒测试　　C. 黑盒测试　　D. 动态测试

【答案】 A

6. 软件验收测试的合格通过准则是(　　)。

A. 软件需求分析说明书中定义的所有功能已全部实现,性能指标全部达到要求

B. 所有测试项没有残余一级、二级和三级错误

C. 立项审批表、需求分析文档、设计文档和编码实现一致

D. 验收测试工件齐全

【答案】 B

7. 软件测试计划评审会需要(　　)参加。

A. 项目经理　　B. SQA 负责人

C. 配置负责人　　D. 测试组

【答案】 A

8. 从测试阶段角度,正确的测试顺序是(　　)。

①单元测试;②集成测试;③系统测试;④验收测试

A. ①②③④　　B. ②①③④　　C. ③②①④　　D. ③①②④

【答案】 A

9. 下列能作为设计阶段测试对象的文档是(　　)。

A. 逻辑设计规格说明　　B. 外部设计规格说明

C. 内部设计规格说明　　D. 以上全对

【答案】 D

10. 一个好的集成测试策略应该具有的特点是(　　)。

A. 能够使模块与接口的划分清晰明了,尽可能减少后续操作难度

B. 能够对被测模块进行比较充分的测试

C. 对整体工作量来说,参加测试的各种资源都得到充分利用

D. 以上全对

【答案】 D

11. 下列说法中错误的是(　　)。

A. 模块在进行集成测试前必须已经通过单元测试

B. 软件集成测试应测试软件单元之间的所有调用

C. 软件集成测试一般采用黑盒测试

D. 软件集成测试应由软件提供方组织实施,不得委托第三方进行测试

【答案】 D

12. 下列测试中能够与软件开发各个阶段(如需求分析、设计、编码)相对应的是(　　)。

A. 组装测试、确认测试、单元测试　　B. 单元测试、组装测试、确认测试

C. 单元测试、确认测试、组装测试　　D. 确认测试、组装测试、单元测试

【答案】 D

13. 单元测试的测试对象是(　　)。

A. 系统　　B. 程序模块　　C. 模块接口　　D. 系统功能

【答案】 B

14. 单元测试时用于代替被调用模块的是(　　)。

A. 桩模块　　B. 通信模块　　C. 驱动模块　　D. 代理模块

【答案】 C

15. (　　)是简化了的模拟较低层次模块功能的虚拟子程序。

A. 过程　　B. 函数　　C. 仿真　　D. 桩

【答案】 D

16. 单元测试的主要任务不包括(　　)。

A. 出错处理　　B. 全局数据结构

C. 独立路径　　D. 模块接口

【答案】 B

17. 单元测试中用来模拟实现被测模块需调用的其他功能模块的是(　　)。

A. 驱动模块　　B. 桩模块

C. 主控模块　　D. 真实的被调用模块

【答案】 B

18. 下列关于 Alpha 测试的描述中正确的是(　　)。

A. Alpha 测试需要用户代表参加　　B. Alpha 测试不需要用户代表参加

C. Alpha 测试是系统测试的一种　　D. Alpha 测试是验收测试的一种

【答案】 A

19. 对于软件的 Beta 测试,下列描述中(　　)是正确的。

A. Beta 测试就是在软件公司内部展开的测试,由公司的专业测试人员执行

B. Beta 测试就是在软件公司内部展开的测试,由公司的非专业测试人员执行

C. Beta 测试就是在软件公司外部展开的测试,由专业测试人员执行

D. Beta 测试就是在软件公司外部展开的测试,由非专业测试人员执行

【答案】 C

20. 软件的单元测试工作通常由(　　)完成。

A. 该软件的设计人员

B. 该软件开发组

C. 不属于该软件开发组的软件设计人员

D. 该软件的编程人员

【答案】 D

21. 单元测试中设计测试用例的依据是(　　)。

A. 概要设计规格说明书　　B. 用户需求规格说明书

C. 项目计划说明书　　D. 详细设计规格说明书

【答案】 D

22. 软件的集成测试工作最好由(　　)承担,以提高集成测试的效果。

A. 该软件的设计人员

B. 不属于该软件开发组的软件设计人员

C. 该软件开发组的负责人

D. 该软件的编程人员

【答案】 B

23. 当对发现的缺陷进行修改之后，执行测试以确认程序的修改没有对程序的其他部分产生干扰。这种测试通常称为(　　)。

A. 验证测试　　B. 回归测试　　C. 系统测试　　D. 确认测试

【答案】 B

24. 下列不属于关键模块具有的特性的是(　　)。

A. 处于程序控制结构的底层　　B. 本身是复杂的或是容易出错的

C. 含有确定的性能需求　　D. 被频繁使用的模块

【答案】 A

25. 在集成测试用例设计的过程中，要满足的基本要求是(　　)。

A. 保证测试用例的正确性

B. 保证测试用例能无误地完成测试项的既定测试目标

C. 保证测试用例的简单性

D. 保证测试用例能满足相应的覆盖率要求

【答案】 C

26. 具有层次结构的大型软件的一种测试方法是从上层模块开始，由上到下进行测试。此时，有必要用一些模块替代尚未测试过的下层模块，这些模块称为(　　)。

A. 桩　　B. 仿真器　　C. 模拟器　　D. 原型

【答案】 A

27. 与设计测试数据无关的文档是(　　)。

A. 该软件的设计文档　　B. 需求规格说明

C. 项目开发计划　　D. 源程序

【答案】 CD

28. 软件单元测试的主要工作内容是(　　)。

A. 测试模块内部逻辑　　B. 测试模块内数据流向

C. 测试模块单元的具体实现　　D. A、B、C

【答案】 D

29. 用来代替被测模块的子模块的是(　　)。

A. 驱动模块　　B. 桩模块　　C. 调用模块　　D. 配置模块

【答案】 B

30. 下列(　　)不是在软件故障插入测试技术中关注的方面。

A. 故障类型　　B. 故障对系统的破坏程度

C. 插入故障的方法　　D. 目标系统

【答案】 B

31. 确认测试应交付的文档主要是(　　)。

A. 确认测试分析报告　　B. 最终的用户手册和操作手册
C. 项目开发总结报告　　D. 以上全部

【答案】 D

32. 除了开发人员之外，首先见到软件产品的人是(　　)。

A. Alpha 测试人员　　B. Beta 测试人员
C. 验收测试人员　　D. 回归测试人员

【答案】 A

33. 验收测试是以(　　)为主的测试。

A. 质量保证人员　　B. 软件开发人员
C. 用户　　D. 软件测试人员

【答案】 C

34. Beta 测试主要衡量产品的 FURPS(功能、易用性、可靠性、性能、支持性)，着重于产品的支持性，包括(　　)。

A. 文档　　B. 客户培训
C. 支持产品生产能力　　D. 以上全部

【答案】 D

35. 下列(　　)不属于回归测试的目的。

A. 检验软件的修改达到了预定目的
B. 检验软件的修改没有影响软件的其他功能的正确性
C. 检验改动没有带来不可预料的行为或者另外的错误
D. 检验修改的测试用例是否完整

【答案】 D

36. 根据软件需求规格说明书，在开发环境下对已经集成的软件系统进行的测试是(　　)。

A. 系统测试　　B. 单元测试　　C. 集成测试　　D. 验收测试

【答案】 A

37. 软件设计阶段的测试主要采取的方式是(　　)。

A. 评审　　B. 白盒测试　　C. 黑盒测试　　D. 动态测试

【答案】 A

38. 集成测试计划应该在(　　)阶段末提交。

A. 需求分析　　B. 概要设计　　C. 详细设计　　D. 单元测试

【答案】 B

39. 将软件组装成系统的测试技术叫(　　)。

A. 集成测试　　B. 单元测试　　C. 集合测试　　D. 系统测试

【答案】 A

40. 集成测试也叫做(　　)。

①单元测试；②部件测试；③组装测试；④系统测试；⑤确认测试；⑥联合测试

A. ③⑥　　B. ①②　　C. ⑤⑥　　D. ③④

【答案】 A

41. 软件测试项目周期是指(　　)。

A. 测试计划　　B. 阶段测试、设计阶段测试、执行阶段

C. 以上都不对　　D. 以上都对

【答案】 D

42. 用户在真实的工作环境中测试软件的用户友好性等,这种测试是(　　)。

A. 集成测试　　B. 系统测试

C. Alpha 测试　　D. Beta 测试

【答案】 D

43. 下列各项都是按照不同阶段对软件测试进行的划分,除了(　　)。

A. 单元测试　　B. 集成测试　　C. 黑盒测试　　D. 系统测试

【答案】 C

44. 测试需求的结构包括(　　)。

A. 需求标识　　B. 需求名称　　C. 需求类型　　D. 优先级

E. 用例关联　　F. 校阅人　　G. 编写日期

【答案】 ABCDEFG

45. 单元测试是对程序设计进行验证,其中(　　)不是单元测试的主要内容。在单元测试过程中,通常测试工程师都需要用(②)代替所测模块调用的子模块。在单元测试的基础上,需要将所有模块按照概要设计和详细设计说明书的要求进度组装,模块组装成系统的方式有两种,分别是(③)。

① A. 模块接口测试　　B. 有效性测试

C. 路径测试　　D. 边界测试

② A. 桩模块　　B. 驱动模块

C. 桩模块和驱动模块　　D. 存根模块和驱动模块

③ A. 一次性组装和增值性组装　　B. 自顶向下组装和自底向上组装

C. 单个模块组装和混合模块组装　　D. 接口组装和功能组装

【答案】 ①B;②A;③A

46. 在执行测试和评价的过程中,会产生较多的文档,(　　)是对文档内容的正确描述。

① 评价需求的主要内容是描述评价的目标,特别是描述了产品的质量需求。

② 评价规格说明的主要内容是确定对软件及其部件实行的所有分析和测量,标识要采用的操作规程、测试方法和工具。

③ 评价记录的主要内容是对评价执行过程的详细记载,由评价请求者保留。

④ 评价报告的主要内容是执行测量和分析的结果,以及能被重复和重新评价的必要信息。

A. ①②　　B. ②③　　C. ①④　　D. ②④

【答案】 C

47. 关于确认测试,以下描述正确的是(　　)。

① 确认测试一般包括有效性测试与软件配置复查，采用黑盒测试为主、白盒测试为辅的方法进行测试。

② 确认测试配置项复查时应当严格检查用户手册和操作手册中规定的使用步骤的完整性和正确性。

③ 确认测试需要检测与证实软件是否满足软件需求说明书中规定的要求。

④ 确认测试是保证软件正确实现特定功能的一系列活动和过程，目的是保证软件生命周期中的每一个阶段的成果满足上一个阶段所设定的目标。

A. ①②　　B. ②③　　C. ③④　　D. ②④

【答案】 B

5.3.2 简答题

1. 软件测试的生命周期是什么？

【答】 软件测试生命周期具体包括以下几个阶段：

(1) 测试计划。根据用户需求报告中关于功能要求和性能指标的规格说明书，定义相应的测试需求报告，选择测试内容，合理安排测试人员、测试时间及测试资源等。

(2) 测试设计。将测试计划阶段制定的测试需求分解、细化为若干个可执行的测试过程，并为每个测试过程选择适当的测试用例，保证测试结果的有效性。

(3) 测试执行。执行测试设计阶段建立的自动测试过程，并对所发现的缺陷进行跟踪管理。测试执行一般由单元测试、组合测试、集成测试以及回归测试等步骤组成。

(4) 测试评估。结合量化的测试覆盖域及缺陷跟踪报告，对于应用软件的质量、开发团队的工作进度及工作效率进行综合评价。

2. 软件测试执行过程有几个阶段？

【答】 软件测试实施一般经历如下 3 个阶段：

(1) 初测期。主要测试软件的主要功能模块和关键的执行路径，排除主要障碍。

(2) 细测期。依据测试计划、测试大纲和测试用例，逐一测试软件的功能、性能、用户界面、兼容性、可用性等多个方面，预期错误的严重程度和问题等。

(3) 回归测试期。此时软件系统在测试中发现的错误十分有限，主要是复查已知错误的纠正情况，当确认未引发任何新的错误时，终结回归测试。

3. 什么是软件测试需求？

【答】 测试需求是制定测试策略、测试用例设计和开发的基础，解决“测什么”的问题，可以从以下几方面来阐释：

(1) 测试需求的核心就是指明被测对象中什么需要测试，包括功能、业务流程、性能等。

(2) 测试需求要全部覆盖已定义的业务流程以及功能和非功能需求。

(3) 制定测试需求项必须是可核实、可观察、可评测的结果。

(4) 测试需求应指明满足需求的正常的前置条件,同时也要指明不满足需求的出错条件。

(5) 测试需求中不涉及测试数据,这属于“怎么测”的问题。

4. 软件测试具体如何执行?

【答】 软件测试的执行包括单元测试、集成测试、确认测试和系统测试。

(1) 单元测试。也称模块测试、逻辑测试、结构测试,测试的方法一般采用白盒测试法,以路径覆盖为测试准则。单元测试属于编码阶段,由程序开发员本人进行,其测试策略包括:设计测试用例要测试哪几方面的问题,针对这几方面的问题各自测试什么内容,测试的具体步骤,实用测试策略。

(2) 集成测试。单元测试之后便进入集成测试,分为增式集成测试和非增式集成测试。

(3) 确认测试。又称合格测试或验收测试。集成测试消除了接口的错误,确认测试由用户参加测试,检验软件规格说明的技术标准的符合程度。

(4) 系统测试。由于软件是数据处理系统中的一个组成部分,软件开发完之后要与系统中的其他部分配合运行,比如将软件、硬件等各部件协调,并对通信等做综合测试。

5. 如何理解单元测试?

【答】 单元测试的对象是程序系统中的最小单元,检查程序模块或组件已实现的功能与定义的功能是否一致,编码中是否存在错误。多个模块可以平行地、对立地测试,通常要编写驱动模块和桩模块。单元测试一般由编程人员和测试人员共同完成。单元测试主要采用白盒测试方法,辅以黑盒测试方法。

6. 单元测试与集成测试有什么区别?

【答】

(1) 测试的单元不同。

单元测试是针对软件的基本单元(如函数)所做的测试,而集成测试则是以模块和子系统为单位进行的测试,主要测试接口间的关系。

(2) 测试的依据不同。

单元测试是针对软件详细设计所做的测试,测试用例主要依据是详细设计。而集成测试是针对高层(概要)设计所做的测试,测试用例主要依据是概要设计。

(3) 测试空间不同。

集成测试的测试空间与单元测试和系统测试是不同的。集成测试不关心内部实现层的测试空间,只关注接口层的测试空间,即关注的是接口层可变数据间的组合关系。

集成测试无法测试从外部输入层的测试空间向接口层测试空间转换时出现的问题,但是可以测试从接口层空间向内部实现层空间进行转换时出现的问题,这是单元测试做不到的。

(4) 具体测试方法不同。

集成测试关注的是接口的集成,和单元测试只关注单个单元不同,因此在具体的测试方法上也不同,集成测试在测试用例设计方面和单元测试有一定的差别。

7. 单元测试与系统测试有什么区别?

【答】 单元测试与系统测试的区别绝不仅仅在于测试的对象和测试的层次不同,更重要的区别是测试的性质不同。单元测试属于白盒测试,关注单元模块内部。单元测试是早期测试,发现问题可以较早地定位。系统测试属于黑盒测试,是站在用户的角度来看待被测系统,该项测试的标准基于客户的需求。系统测试是后期测试,发现错误后的定位工作比较困难。

8. 简述集成测试和系统测试有什么区别?

【答】 集成测试的主要依据是概要设计说明书,系统测试的主要依据是需求设计说明书。集成测试是系统模块的测试;系统测试是对整个系统的测试,包括相关的软硬件平台,网络及相关的外设的测试。

9. 在集成测试时需要考虑哪些问题?

【答】

(1) 在把各个模块连接起来的时候,穿越模块接口的数据是否会丢失。

(2) 一个模块的功能是否会对另一个模块的功能产生不利的影响。

(3) 各个子功能组合起来后能否达到预期要求的父功能。

(4) 全局数据结构是否有问题。

(5) 单个模块的误差累积起来是否会放大,以至达到不能接受的程度。

10. 集成测试的原则是什么?

【答】

(1) 所有公共接口必须被测试到。

(2) 关键模块必须进行充分测试。

(3) 集成测试应当按一定层次进行。

(4) 集成测试策略选择应当综合考虑质量、成本和进度三者之间的关系。

(5) 集成测试应当尽早开始,并以概要设计为基础。

(6) 在模块和接口的划分上,测试人员应该和开发人员充分沟通。

(7) 当测试计划中的结束标准满足时,集成测试才能结束。

(8) 当接口发生修改时,涉及的相关接口都必须进行回归测试。

(9) 集成测试应根据集成测试计划和方案进行,不能随意测试。

(10) 项目管理者应保证测试用例经过审核。

(11) 测试执行结果应当如实地记录。

11. 集成测试策略主要有哪些？

【答】

（1）大爆炸集成。又称一次性组装或整体拼装，属于非增值式集成。这种集成策略的做法就是把所有通过单元测试的模块一次性集成到一起进行测试，不考虑组件之间的互相依赖性及可能存在的风险。

（2）三明治集成。一种混合增量式测试策略，综合了自顶向下和自底向上两种集成方法的优点，因此也属于基于功能分解的集成。采用这种方法，桩和开发工作都比较小，但增加了定位缺陷的难度。

（3）自顶向下集成。就是按照系统层次结构图，以主程序模块为中心，自上而下按照深度优先或者广度优先策略，对各个模块一边组装一边进行测试。又可分为深度优先集成和广度优先集成两种方式。

（4）自底向上集成。从依赖性最小的底层模块开始，按照层次结构图逐层向上集成，验证系统的稳定性。

（5）高频集成。与软件开发过程同步，每隔一段时间对开发团队的现有代码进行一次集成测试。

12. 采用自顶向下的增值方式进行组装测试，其步骤是什么？

【答】

（1）所有直属于主模块的下属模块全部用桩模块代替，对主模块进行测试。

（2）采用深度优先或分层的策略，用实际模块替换相应的桩模块，再用桩模块代替它们的直接下属模块，与已测试的模块或子系统组装成新的子系统。

（3）进行回归测试（即重新执行以前做过的全部测试或部分测试），排除引入新的错误的可能。

（4）判断是否所有的模块都已组装到系统中，是则结束测试，否则转到（2）执行。

13. 采用混合增值式测试时，有3种常见的综合的增值方式是什么？

【答】

（1）演变的自顶向下的增值测试。用于对输入输出模块和引入新算法模块的测试，并自底向上组装成为功能相当完整且相对独立的子系统，然后由主模块开始自顶向下进行增值测试。

（2）自底向上-自顶向下的增值测试。首先对含读操作的子系统自底向上至根结点模块进行组装和测试，然后对含写操作的子系统做自顶向下的组装与测试。

（3）回归测试。这种方法采用自顶向下的方式测试所修改的模块及其子模块，然后将这一部分视为子系统，再自底向上地测试，以检查该子系统与其上级模块的接口是否适配。

14. 非渐增式测试与渐增式测试有什么区别?

【答】

(1) 非渐增式测试把单元测试和集成测试分成两个不同的阶段,前一阶段完成模块的单元测试,后一阶段完成集成测试。而渐增式测试往往把单元测试和集成测试合在一起,同时完成。

(2) 非渐增式测试需要更多的工作量,因为每个模块都需要驱动模块和桩模块;而渐增式测试利用已测试过的模块作为驱动模块或桩模块,因此工作量少。

(3) 渐增式测试可以较早地发现接口之间的错误,非渐增式测试最后组装时才能发现接口错误。

(4) 渐增式测试有利于排错,发生错误往往和最近新加入的模块有关;而非渐增式测试发现接口错误推迟到最后,很难判断是哪一部分接口出错。

15. 软件测试各个阶段与软件开发各阶段之间有什么关系?

【答】 软件测试要经过的步骤是单元测试→集成测试→确认测试→系统测试。

单元测试对源程序中每一个程序单元进行测试,检查各个模块是否正确实现规定的功能,从而发现模块在编码中或算法中的错误。该阶段涉及编码和详细设计文档。

集成测试是为了检查与设计相关的软件体系结构的有关问题,也就是检查概要设计是否合理、有效。

确认测试主要是检查已实现的软件是否满足需求规格说明书中确定了的各种需求。

系统测试是把已确认的软件与其他系统元素(如硬件、其他支持软件、数据、人工等)结合在一起进行测试,以确定软件是否可以交付使用。

16. Alpha 测试与 Beta 测试的区别是什么?

【答】 Alpha 测试和 Beta 测试用于发现可能只有最终用户才能发现的错误。Alpha 测试是在开发环境下或者公司内部的用户在模拟实际操作环境下,由用户参与的测试。其测试目的主要是评价软件产品的功能、可使用性、可靠性、性能等,特别是对于软件的界面和使用方式的测试。Beta 测试是在实际使用环境下进行的测试,与 Alpha 测试不同,开发者通常不在测试现场。因而,Beta 测试是在开发者无法控制的环境下进行的软件现场应用。在 Beta 测试中,由用户记下遇到的所有问题,包括真实的以及主观认定的,定期向开发者报告。开发者在综合用户的报告之后,做出修改,最后将软件产品交付给全体用户使用。Beta 测试着重于产品的支持性,包括文档、客户培训和支持产品生产能力。只有当 Alpha 测试达到一定的可靠程度时,才能开始 Beta 测试。

17. 常用的回归测试用例有几种选择方法?

【答】 常用的回归测试用例选择有如下几种方法。

(1) 在修改范围内的测试。这类回归测试仅根据修改的内容来选择测试用例,仅保证修改的缺陷或新增的功能被实现。这种方法的效率最高,然而风险也最大,因为它无法

保证这个修改不影响别的功能。该方法一般用于软件结构设计的耦合度较小的情况。

(2) 在受影响范围内回归。这类回归测试需要分析可能影响的代码或功能,对于所有受影响的功能和代码,其对应的所有测试用例都将被回归。判断哪些功能或代码受影响,往往依赖于测试人员的经验和开发过程的规范性。

(3) 根据一定的覆盖率指标选择回归测试。例如,规定修改范围内的测试覆盖率是90%,其他范围内的测试覆盖率阈值为60%,该方法一般在相关功能影响范围难以界定时使用。

(4) 基于操作剖面选择测试。如果测试用例是基于软件操作剖面开发的,测试用例的分布情况将反映系统的实际使用情况。回归测试所使用的测试用例个数由测试预算确定,可以优先选择针对最重要或最频繁使用功能的测试用例,尽早发现对可靠性有最大影响的故障。

(5) 基于风险选择测试。根据缺陷的严重性来进行测试,基于一定的风险标准从测试用例库中选择回归测试包。选择关键以及可疑的测试,跳过那些次要的、例外的测试用例或功能非常稳定的模块。

总之,应依据经验和判断选择不同的回归测试技术和方法,综合运用多种测试技术。

18. 驱动模块和桩模块分别是什么?

【答】 驱动模型也称驱动程序(driver),是对底层或子层模块进行(单元或集成)测试时所编制的调用被测模块的程序,用以模拟被测模块的上级模块。桩模块也称桩程序(stub)也称存根程序,是对顶层或上层模块进行测试时,所编制的替代下层模块的程序,用以模拟被测模块工作过程中所调用的模块,如图5.1所示。

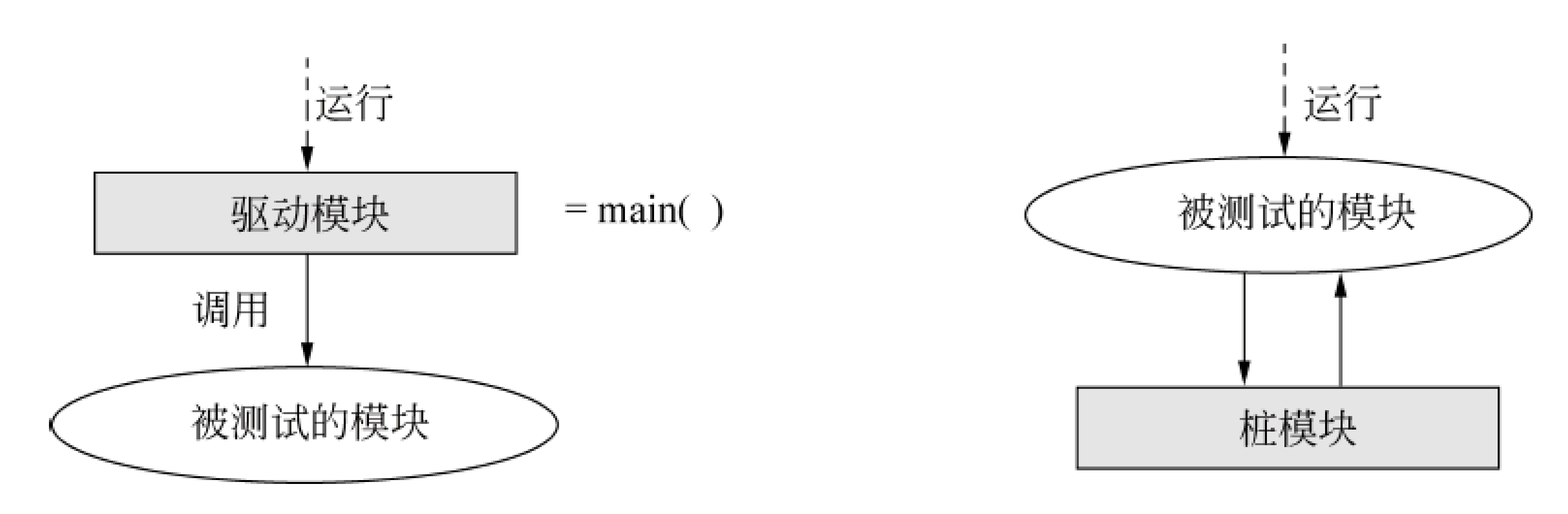


图5.1 驱动模块和桩模块

19. 如图5.2所示,B模块为被测模块,请给出示意图说明如何添加驱动模块和桩模块。

【答】 添加驱动模块和桩模块后,被测模块B的测试环境如图5.3所示。

20. 系统测试与单元测试、集成测试有哪些区别?

【答】

(1) 测试方法不同。系统测试主要是黑盒测试,而单元测试、集成测试主要属于白盒测试范畴。

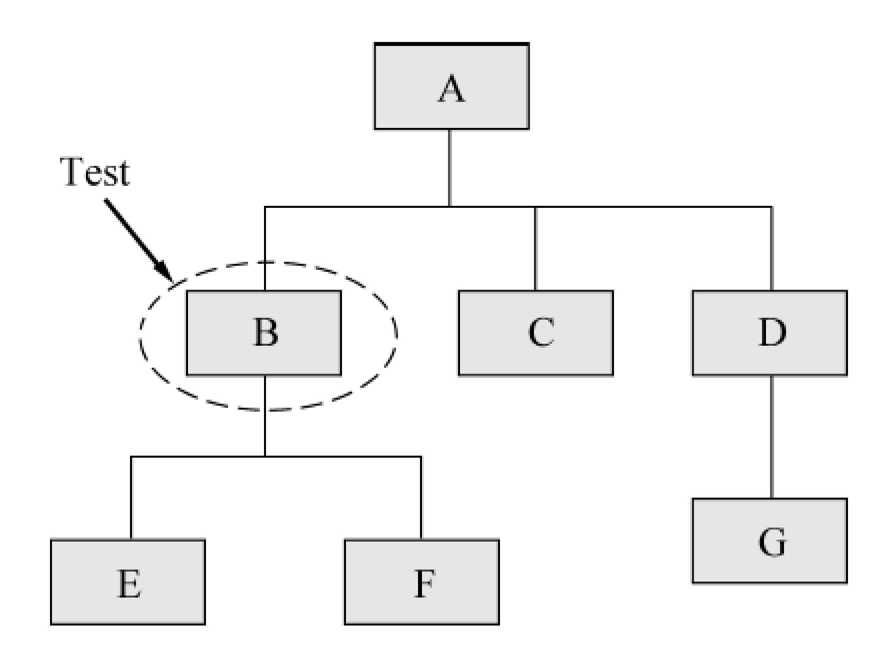


图 5.2　B 模块为被测模块

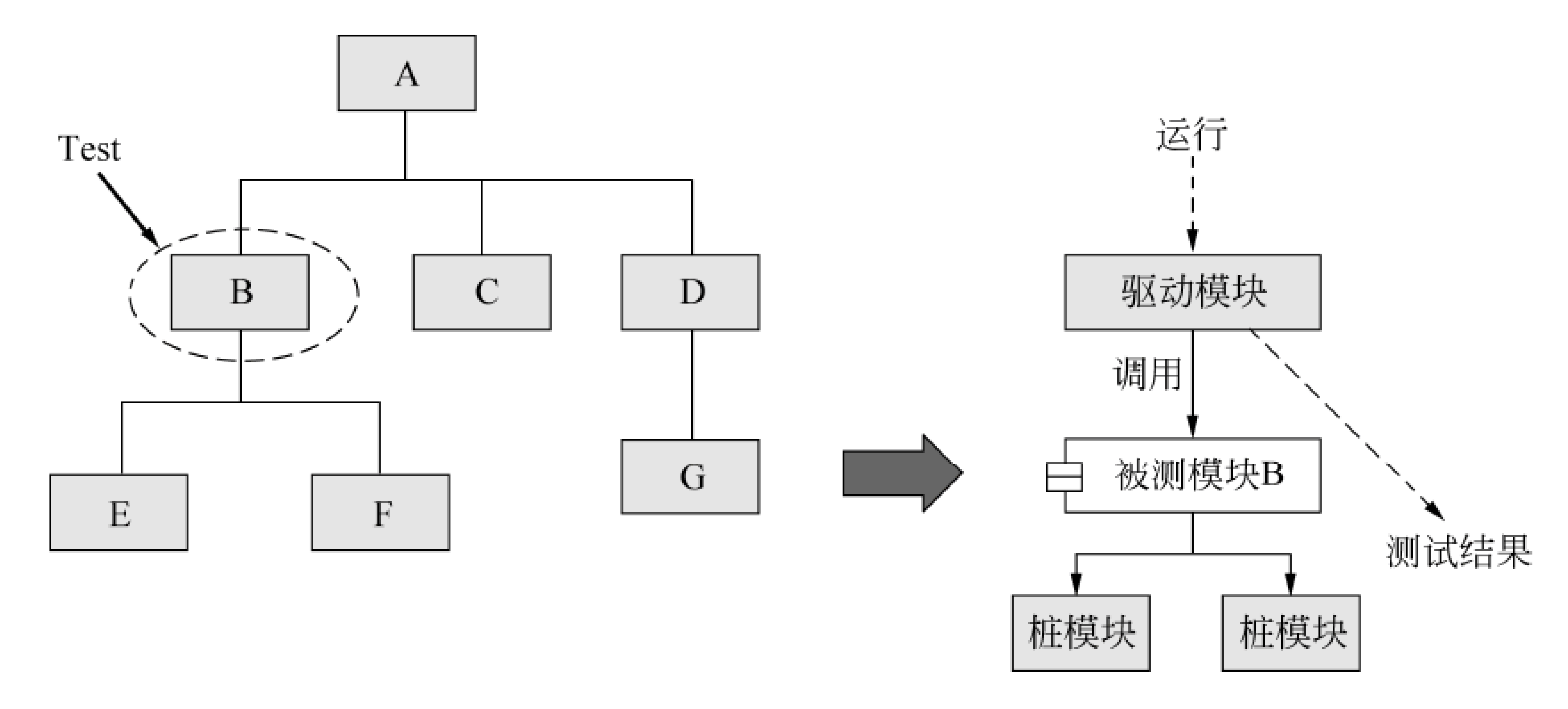
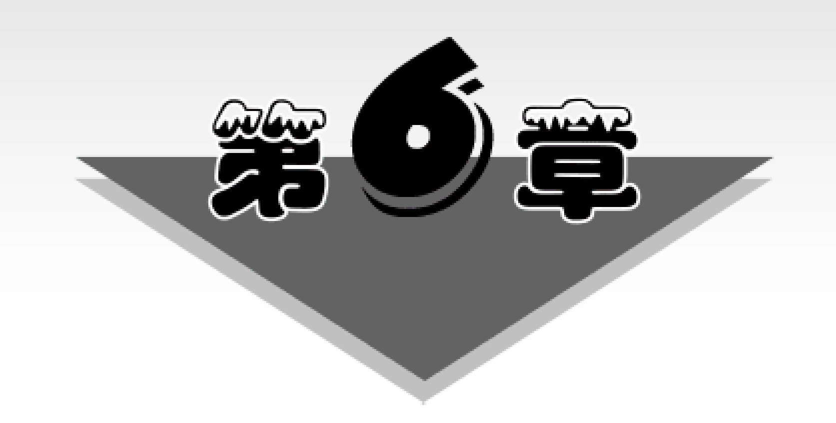


图 5.3　驱动模块和桩模块

(2) 考察范围不同。单元测试主要是测试模块内部的接口、数据结构、逻辑、异常处理等,集成测试主要测试模块间的接口和异常,系统测试主要测试整个系统相对于用户的需求。

(3) 评估基础不同。系统测试的评估基准是测试用例对需求规格说明的覆盖率,而单元测试和集成测试的评估主要是代码的覆盖率。



性 能 测 试

6.1 本 章 要 求

- 了解性能指标。
- 理解性能测试分类。
- 了解 Web 测试。

6.2 本章知识重点

1. 性能测试

性能测试主要包括以下几个方面：

（1）评估系统的能力。通过测试软件系统中的负荷和响应时间数据来验证软件系统的稳定性和可靠性。

（2）识别体系中的弱点。通过将软件系统受控的负荷增加到极端的水平，确定体系的瓶颈或薄弱的地方，并进行修复。

（3）系统调优。通过长时间地运行系统，导致系统失败，以揭示系统中的隐含的问题或冲突，优化系统性能。

2. 常见的性能指标

1）响应时间

响应时间是指“对请求做出响应所需要的时间”，分解为网络传输时间、应用延迟时间、数据库延迟时间和呈现时间等。其中，“呈现时间”取决于数据在被客户端收到数据后呈现页面所消耗的时间，“系统响应时间”指应用系统从请求发出开始到客户端接收到数据所消耗的时间。一般而言，网站的响应时间有 2 秒、5 秒和 10 秒几个标准。2 秒之内响应客户被认为是“非常有吸引力的”，5 秒之内响应客户认为是“比较不错的”，而 10 秒是客户能接受的响应的上限，也就是用户打开网页花费时间超过了 10 秒钟，用户往往无法容忍。

2）并发用户数

多个用户对系统发出了请求或者进行了操作，其请求或者操作可以是相同的，也可以

是不同的。下面给出估算并发用户数的公式：

$$C = \frac{nL}{T} \tag{6.1}$$

$$\hat{C} \approx C + 3\sqrt{C} \tag{6.2}$$

式(6.1)中，C 是平均的并发用户数，n 是登录会话的数量，L 是登录会话的平均长度，T 指考察的时间段长度。式(6.2)给出了并发用户数峰值的计算方式，其中，$\hat{C}$ 指并发用户数的峰值，C 由式(6.1)得到，式(6.2)是假设用户登录会话符合泊松分布而估算得到的。

3）吞吐量

吞吐量是指在单位时间内成功地在网络上传输数据量的总和。一般来说，吞吐量用每秒请求数或每秒页面数来衡量。吐量指标有如下两个作用：

(1) 协助设计性能测试场景，以及衡量性能测试场景是否达到了预期的设计目标。在设计性能测试场景时，根据吞吐量数据测试场景的事务发生频率等。

(2) 协助分析性能瓶颈。吞吐量是性能瓶颈的重要表现形式。因此，有针对性地测试吞吐量，可以尽快定位到性能瓶颈所在位置。

吞吐量和并发用户数之间存在一定的联系。计算公式如下：

$$F = \frac{N_{vu} \times R}{T} \tag{6.3}$$

其中，F 表示吞吐量，N_{vu}表示虚拟用户个数，R 表示每个虚拟用户发出的请求数量，T 表示测试时间。

4）性能计数器

性能计数器是描述服务器或操作系统性能的一些数据指标，具有监控和分析作用。例如，Windows 系统的内存数、进程数、系统缓存等都是常见的性能计数器。与性能计数器相关的资源利用率是指系统各种资源的使用状况，计算公式如下：

资源利用率＝资源的实际使用/总的资源可用量

通常的情况下，资源利用率需要结合响应时间变化曲线、系统负载曲线等各种指标进行综合分析。

6.3 典型习题解析

6.3.1 选择题

1. 以消除瓶颈为目的的测试是(　　)。

A. 负载测试　　B. 性能测试　　C. 动态测试　　D. 覆盖测试

【答案】 A

2. 软件可靠性的(　　)特性是指在软件发生故障的情况下，软件产品维持规定的性能级别的能力。

A. 成熟性　　B. 易恢复性　　C. 容错性　　D. 可靠性

【答案】 C

3. 下列测试中不属于系统测试的是(　　)。

A. 性能测试　B. 集成测试　C. 压力测试　D. 可靠性测试

【答案】 B

4. 下面的说法中属于系统测试目标的是(　　)。

① 找出软件中存在的缺陷与错误。

② 确认软件所实现的功能是否符合规格说明。

③ 确认软件的性能是否满足要求。

A. ①　B. ①②　C. ②③　D. ①②③

【答案】 D

5. (　　)在系统测试分析阶段无须考虑。

A. 用户层　B. 应用层　C. 数据层　D. 协议层

【答案】 C

6. 以下选项中(　　)不属于协议一致性测试分析方法。

A. 基本互联测试　B. 定向诊断测试　C. 人员测试　D. 能力测试

【答案】 C

7. 在性能测试中，关于数据准备，(　　)的描述是正确的。

① 识别数据状态验证测试案例。

② 初始数据提供了基线用来评估测试执行的结果。

③ 业务数据提供负载压力背景。

④ 脚本中参数数据真实模拟负载。

A. ①②③　B. ①③④　C. ②③　D. ①②③④

【答案】 D

8. 不属于界面元素测试的是(　　)。

A. 窗口测试　B. 文字测试　C. 功能点测试　D. 鼠标测试

【答案】 C

9. 以下说法不正确的是(　　)。

A. 易用性测试不仅是针对应用程序的测试，而且还要包括用户手册等系列文档

B. 安装测试就是按照用户安装手册安装软件，以评估安装过程的易用性、正确性

C. 辅助系统测试包括帮助测试、向导测试、信息提示测试等

D. 界面整体测试是指对界面的规范性、可维护性、整体性等进行测试和评估

【答案】 D

10. 负载压力性能测试需求分析时，应该选择(　　)类型的业务作为测试案例。

①高吞吐量的业务；②业务逻辑复杂的业务；③高商业风险的业务；④高服务器负载的业务；⑤批处理的业务。

A. ①②③　B. ①③④　C. ①④　D. ①②③④⑤

【答案】 B

11. 以下描述中正确的有(　　)。

① 响应时间是指从按动传送键到得到结果为止所需要的时间。
② 处理时间是指计算机从接收一个消息到送出它的结果所经过的时间。
③ 周转时间是指从提出要求到得到结果所需要的时间。
④ 响应时间包括处理时间和传输时间。

A. ①②③④　　B. ①③　　C. ②③　　D. ①②④

【答案】 A

12. 系统测试是将已经集成好的软件系统与其他系统元素结合在一起，进行一系列的(　　)。

A. 单元测试和集成测试　　B. 单元测试、集成测试和确认测试
C. 集成测试和确认测试　　D. 验收测试

【答案】 C

13. 系统测试中最基本的测试策略是(　　)。

A. 功能测试　　B. 性能测试　　C. 安全性测试　　D. 压力测试

【答案】 A

14. 以下关于系统测试方法的说法不正确的是(　　)。

A. 可以使用监视器方法收集系统执行时间和资源使用情况
B. 只要有足够的时间，一个好的安全测试就一定可以侵入一个系统
C. 容量测试是指系统承受速度方面的超额负载
D. 在嵌入式系统中，功能需求与性能需求必须同时考虑

【答案】 C

15. 在进行健壮性测试时，常用的测试用例设计方法主要有(　　)。

A. 故障插入测试　　B. 安全性测试
C. 变异测试　　D. 错误猜测法

【答案】 ACD

16. 下列不属于安全性的性能是(　　)。

A. 有效性　　B. 生存性　　C. 一致性　　D. 精确性

【答案】 C

17. 在 Web 应用软件的分层测试策略中，下列(　　)不是测试关注的层次。

A. 数据层　　B. 业务层　　C. 服务层　　D. 表示层

【答案】 C

6.3.2 简答题

1. 系统测试是什么？

【答】 用户的需求分为功能性需求和非功能性需求，非功能性需求一般被归纳为软件产品的各种质量特性，如安全性、兼容性和可靠性等。系统测试就是针对这些非功能特性展开的，即验证软件产品符合这些质量特性的要求，从而满足用户和软件企业自身的非功能性需求。所以，系统测试分为负载测试、性能系统、容量测试、安全性测试、兼容性测

试和可靠性测试等。

2. **负载测试与压力测试有什么异同点？**

【答】 压力测试可以被看作是负载测试的一种，即高负载下的负载测试。压力测试是在系统（如 CPU、内存和网络带宽等）处于饱和状态下，测试系统是否还具有正常的会话能力、数据处理能力或是否会出现错误，以检查软件系统对异常情况的抵抗能力，找出性能瓶颈、功能不稳定性等问题。

压力测试分为稳定性压力测试和破坏性压力测试。稳定性压力测是指高负载下持续运行 24 小时以上的压力测试。而破坏性压力测试是通过不断加载的手段快速造成系统崩溃，让问题尽快地暴露出来。

3. **压力测试具有哪些特点？**

【答】

(1) 压力测试是检查系统处于压力情况下的能力表现，比如，通过增加并发用户的数量检测系统的服务能力和水平，通过增加文件记录数来检测数据处理的能力和水平，等等。

(2) 压力测试一般通过模拟方法进行。通常在系统对内存和 CPU 利用率进行模拟，以获得测量结果。如将压力的基准设定为内存使用率达到 75%以上，CPU 使用率达到 75%以上，并在此观测系统响应时间、系统有无错误产生。此外，数据库的连接数量、数据库服务器的 CPU 利用率等也都可以作为压力测试的依据。

(3) 压力测试一般用于测试系统的稳定性。测试系统是否能在压力环境下稳定运行一段时间，在压力测试中，通常会考察系统在压力下是否会出现错误等问题。

4. **请解释容错性、成熟性和易恢复性。**

【答】 容错性是指在软件发生故障或者违反指定接口的情况下，软件产品维持规定的性能级别的能力。成熟性是指软件产品避免因软件中错误的发生而导致失效的能力。易恢复性是指在失效发生的情况下，软件产品重建规定的性能级别并恢复受直接影响的数据的能力。

5. **性能测试与容量测试有什么异同点？**

【答】 性能测试是为获取或验证系统性能指标而进行的测试。容量测试/通过负载测试或其他测试方法预先分析出反映软件系统应用特征的某项指标的极限值（如最大并发用户数、数据库记录数等），在其极限值状态下系统主要功能还能保持正常运行。

容量测试属于性能测试中的一种，一般采用逐步加载的负载测试方法，也可以先采用逐步加载方式，获得一个基本的容量值或容量范围，然后再考虑用一次性加载方式来决定实际可支持的容量值。

6. 容量测试与压力测试的区别是什么?

【答】 与容量测试十分相近的概念是压力测试,二者都是检测系统在特定情况下能够承担的极限值。然而两者的侧重点有所不同,压力测试主要是使系统承受速度方面的超额负载,例如一个短时间之内的吞吐量。容量测试关注的是数据方面的承受能力,并且它的目的是显示系统可以处理的数据容量。

7. 什么是健壮性测试?

【答】 健壮性测试主要用于测试系统抵御错误的能力。这里的错误通常指的是由于设计缺陷而带来的系统错误。测试的重点为当出现故障时是否能够自动恢复或忽略故障,继续运行。

8. 什么是验收测试?

【答】 验收测试是按照项目任务书、合同或供需双方约定的验收依据文档对整个系统进行测试与评审,决定是否接收或拒收系统。在系统测试的后期,以用户测试为主或有测试人员等质量保证人员共同参与的验收测试的目的是向未来的用户表明系统能够像预定要求那样工作,验证软件的功能和性能与用户所合理期待的一致。

9. 什么是安装测试?

【答】 安装测试是指按照软件产品安装手册或相应的文档,在一个和用户使用该产品完全一样的环境中或与之相当的环境中进行一步一步的安装操作性的测试。

10. 什么是安全性测试? 它与可靠性测试有什么区别?

【答】 安全性测试是测试系统在应付非授权的内部/外部访问、非法侵入或故意的损坏时的系统防护能力,检验系统有能力使可能存在的内部/外部的伤害或损害的风险限制在可接受的水平。可靠性通常包括安全性,但是软件的可靠性不能完全取代软件的安全性,安全性还涉及数据加密、保密、存取权限等多个方面。

进行安全性测试时需要设计一些测试用例尝试突破系统的安全保密措施,检验系统是否有安全保密漏洞,验证系统的保护机制是否能够在实际中不受非法侵入。安全性测试要建立整体的威胁模型,测试溢出漏洞、信息泄漏、错误处理、SQL 注入、身份验证和授权错误、XSS 攻击。在安全测试过程中,测试者扮演试图攻击系统的角色设计测试用例。例如:(1)尝试截取、破译、获取系统密码;(2)让系统失效、瘫痪,将系统制服,使他人无法访问,自己非法进入。

11. 一般会从哪些方面来判断软件的可靠性?

【答】

(1) 用户权限限制。软件是否按功能模块划分用户权限,权限划分是否合理,考察超级用户对各个用户的权限管理是否合理,包括修改用户的登录资料等。

(2) 用户和密码封闭性。软件对用户名和密码有无校验，有无保护措施，尤其对密码有无屏蔽功能。

(3) 系统对用户错误登录的次数限制。软件对用户错误登录有无次数限制，一般做法是连续3次登录失败就退出系统。

(4) 留痕功能。软件是否提供操作日志，比如某用户登录的时间，查询、修改或删除以及离开的时间等。

(5) 屏蔽用户操作错误。考察对用户常见的误操作的提示和屏蔽情况，例如可否有效避免日期的录入错误或写入无效的日期。

(6) 错误提示的准确性。当用户操作错误或软件发生错误时，能否给出准确清晰的提示，使用户知道造成错误的原因。例如，当用户未输入完有效信息时存盘，系统应当给出关于未输入项的提示。

(7) 错误是否导致系统异常退出。考察软件运行的稳定性，当软件发生一般错误或严重错误时，软件是否会自动退出。

(8) 数据备份与恢复手段。主要针对有数据存储需要的软件，有的软件依靠数据库管理系统本身的备份与恢复机制，这需要用户具备一定的数据库操作知识。好的软件会提供备份与恢复的操作，不需要用户直接对数据库系统进行操作。

(9) 输入数据有效性检查。当用户输入的数据有错时，软件应能判断数据的有效性，避免无效数据的生成。

(10) 异常情况的影响。在程序运行过程中进行掉电等试验，考察数据和系统的受影响程度；若受损，软件是否提供补救工具，补救的情况如何。

(11) 网络故障对系统的影响。当网络中断连接时，是否会造成数据的丢失。

12. 请简要说明Web测试应注意的问题。

【答】 Web测试大致可分为6个部分：

(1) 用户界面测试。要注意是否有使用说明、站点地图和导航条，关注内容、颜色/背景、图片表格等。

(2) 功能测试。要关注链接、信息交互、数据校验等。

(3) 接口测试。要关注服务器接口、外部接口、错误处理等。

(4) 兼容性测试。要关注操作系统、浏览器、Modem/连接速率、硬件设备等的兼容性。

(5) 负载/压力测试。要关注瞬间访问高峰、每个用户传送大量数据、长时间的使用等。

(6) 安全测试。要关注目录设置、登录、日志文件等。

13. 配置测试环境一般需遵循哪些原则？

【答】

(1) 符合软件运行的最低要求。测试环境首先要保证能支撑软件正常运行。

(2) 选用比较普及的操作系统和软件平台。

（3）营造相对简单、独立的测试环境。除了操作系统，测试机上只安装软件运行和测试必需的软件，以免不相关的软件影响测试实施。

（4）无毒的环境。利用有效的正版杀毒软件检测软件环境，保证测试环境中没有病毒。

14. 恢复性测试和备份测试是什么关系？

【答】 恢复性测试主要检查系统的容错能力，即，当系统出错时，能否在指定时间间隔内修正错误并重新启动系统。备份测试是恢复性测试的一个补充，也是恢复性测试的一个部分。备份测试的目的是验证系统在软件或者硬件失败时备份数据的能力。

15. 用户文档的测试一般要关注文档的哪些特性？

【答】

（1）用户文档的完整性。用户文档应包含产品使用所需要的全部信息。

（2）用户文档的正确性。用户文档中所有信息应是正确的，不能有歧义和错误的表达。

（3）用户文档的一致性。用户文档自身内容或相互之间以及与软件系统之间都不应相互矛盾。

（4）用户文档的易理解性。用户文档对于正常执行其工作任务的一般用户应是易理解的。

（5）用户文档的易浏览性。用户文档易于浏览，相互关系明确，每个文档有目录和索引表。

16. 兼容性测试是什么？

【答】 兼容性测试是指检查软件之间是否能够正确地进行交互和共享信息。对新软件进行软件兼容性测试，需要解决以下问题：

（1）软件设计要求与何种其他平台和应用软件保持兼容？如果要测试的软件是一个平台，那么软件设计要求什么应用程序在其上运行？

（2）软件使用何种数据与其他平台和软件交互和共享信息？

17. 什么是可用性测试？

【答】 可用性测试是对于用户友好性的测试，是指在软件测试中用来改善易用性的一系列方法。

18. 文档测试是什么？

【答】 文档测试是指对软件开发和测试维护过程中产生的所有文档的测试，包括对需求规格分析说明书、详细设计报告、系统设计报告、用户手册以及与系统相关的文档的审阅和评测。

19. Web应用的测试策略是什么？

【答】 Web 应用的测试策略如表 6.1 所示。

表 6.1 Web 应用的测试策略

类别	细分	描述
功能和结构测试	导航和链接测试	测试所有链接是否像指示的那样确实链接到了该链接的页面； 测试所链接的页面是否存在； 确保 Web 应用系统中没有孤立的页面； 测试动态页面根据不同的动态条件构建的静态页面符合需要； 导航是否直观； Web 页面的主要部分是否可以通过主页到达； Web 应用是否需要导航地图的帮助
	状态行为测试	测试页面之间的依赖关系是否保持
	功能模块测试	测试各个功能单元是否完成了预定的功能； 测试数据的一致性和输出错误等； 提交操作的完整性，校验默认值的正确性
	数据流测试 信息流测试	测试页面或者页面中使用的脚本、调用的类中的变量是否被正确地定义和使用等
统计测试	统计测试	测试关键功能和关键页面，使测试效率达到最高
性能测试	负载测试	测试 Web 应用在某一负载级别上的性能，确保 Web 应用在需求范围内正常工作
	强度测试	测试 Web 应用的极限和故障恢复能力
	持久度测试	测试 Web 应用在较长时间段内的性能，发现内存泄漏等难以发现的问题
可用性测试	整体界面和内容测试	图形有明确的用途；验证页面字体风格是否一致；图片的大小和质量
		Web 应用的页面所显示的内容的正确性、完整性和相关性
		整体界面是否风格较为一致
兼容性测试	平台、浏览器兼容性测试	测试不同操作系统平台、不同浏览器版本下，页面是否都能正确地显示出来
安全性测试		登录测试；是否有超时的限制；需要测试的相关信息是否写入了日志； 需要加密的信息是否经过了加密后存储或者传输；测试没有经过授权，是否可以访问不能访问的功能
大数据量测试		测试大数据量的情况下 Web 应用是否正常地完成了预定的功能

6.3.3 设计题

1. 一个软件系统每天大约有 800 个用户访问。用户在一天的 8 个小时内使用该系统，从登录到退出该系统的平均时间为 4 小时。请问并发用户数的峰值是多少。

【解析】 根据式(6.1)和式(6.2),计算如下:

$$C=\frac{nL}{T}=800\times 4/8=400$$

$$\hat{C}\approx C+3\sqrt{C}=400+3\times\sqrt{400}=460$$

2. 某系统投入测试,工作一段时间 t_1 后,软件出现错误,系统被停止。修复花了 T_1 时间,故障被排除,又投入测试或运行。假设 $t_1,t_2,\cdots,t_n$ 是系统正常的工作时间,T_1,$T_2,\cdots,T_n$ 是修复时间,如图 6.1 所示。图 6.2 是系统的实际工作与修复时间的示意图,其中,阴影部分代表修复时间。

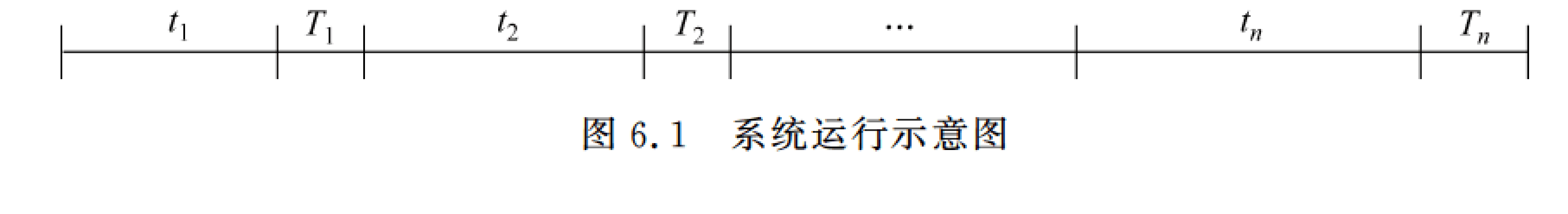


图 6.1 系统运行示意图



图 6.2 系统实际工作时间和修复时间示意图

请计算平均无故障时间(MTBF)、故障率(风险函数,λ)、平均维护时间(MTTR)、维修率(μ)、有效度(A)。

【解析】

$$n=6$$

无故障天数为

$$15+6+20+30+40+55+20=186$$

$$t=186\times 8=1488(\text{每天 8h})$$

$$T=2+3+2.5+2.5+2.5+2.5=15\text{h}$$

平均无故障时间(MTBF)等于总工作时间除以总失效次数:

$$\text{MTBF}=\frac{\sum_{i=1}^{n}t_i}{n}=\frac{1488}{6}=248\text{h}$$

故障率(风险函数)λ等于总失效次数除以总工作时间:

$$\lambda=\frac{n}{\sum_{i=1}^{n}t_i}=\frac{6}{1488}\approx 0.004$$

平均维护时间(MTTR)等于总维护时间除以总失效次数:

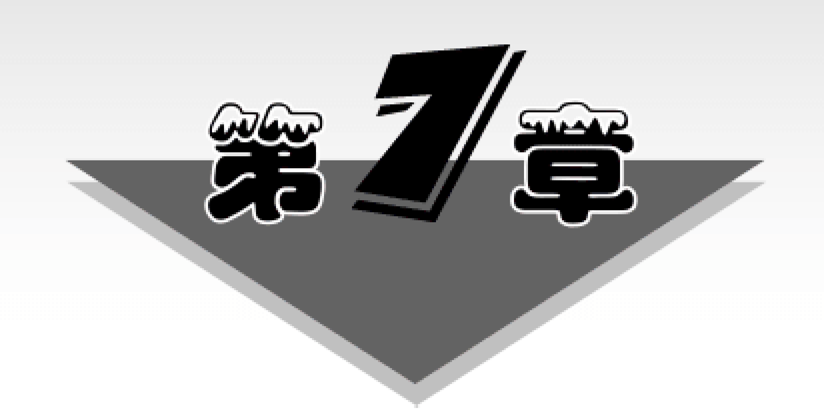
$$\text{MTTR}=\frac{\sum_{i=1}^{n}T_i}{n}=\frac{15}{6}=2.5\text{h}$$

维修率(μ)等于总失效次数除以总维护时间:

$$\mu = \frac{n}{\sum_{i=1}^{n} T_i} = \frac{6}{15} = 0.4$$

有效度(A)等于总工作时间除以(总工作时间＋总维护时间)：

$$A = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}} = \frac{\mu}{\lambda + \mu} = \frac{0.4}{0.004 + 0.4} \approx 0.99$$



自动测试技术

7.1 本章要求

- 了解自动化测试的作用及应用场合。
- 了解测试成熟度模型。
- 了解自动化测试原理。
- 了解自动化测试工具分类、选择特征。
- 掌握相关测试软件的使用。

7.2 本章知识重点

自动化测试把以人为驱动的测试行为转化为机器执行的一种过程。软件测试工具能够实现手工测试无法实现的功能，减轻手工测试的工作量，减少测试的执行时间，提高测试效率。

自动化测试适合以下场合：

（1）软件需求变动不频繁。当软件需求变动过于频繁时，势必多次更新测试用例以及测试脚本，增大维护脚本的成本。一般情况下，对于需求中相对稳定的模块进行自动化测试，变动较大的模块采用手工测试。

（2）项目周期足够长。自动化测试需求的确定、自动化测试框架的设计、测试脚本的编写与调试需要相当长的时间来完成。

（3）自动化测试脚本可重复使用，手工测试完成难度较大的场合。性能测试、压力测试、负载测试等需要模拟大量并发用户时，自动化测试脚本可重复使用。许多与时序、死锁、资源冲突、多线程等有关的软件缺陷很难通过手工测试发现。

（4）回归测试。回归测试是软件每次有新版本时都必须执行，也就是在软件的生命周期中会被反复执行的测试，因此这类测试很适合采用自动化测试。

当然，软件测试工具也有以下不足：

（1）某些测试工具难于学习和使用，创建和修改测试脚本费时费力。

（2）根据实际需要确定是否选用测试工具和选用什么样的测试工具，测试工具往往只能解决某一方面问题，应用范围狭窄。

1. 测试成熟度模型

测试成熟度模型(Testing Maturity Model,TMM)描述了测试的过程,分为初始级、定义级、集成级、管理和度量级、优化级5个等级,如表7.1所示。

表7.1 测试成熟度模型的基本描述

级别	简单描述	特征	目标
初始级	测试处于混乱的状态,缺乏成熟的测试目标,处于可有可无的地位	还不能把测试同调试分开;编码完成后才进行测试工作;测试的目的是表明程序没有错误;缺乏相应的测试资源	
定义级	测试目标是验证软件符合需求,会采用基本的测试技术和方法	测试被看做是有计划的活动;测试同调试分开;在编码完成后才进行测试工作	启动测试计划过程;将基本的测试技术和方法制度化
集成级	测试不再是编码后的一个阶段,而是贯穿在整个软件生命周期中,测试建立在满足用户或客户的需求上	具有独立的测试部门;根据用户需求设计测试用例;有测试工具辅助进行测试工作;没有建立起有效的评审制度;没有建立起质量控制和质量度量标准	建立软件测试组织;制订技术培训计划;测试在整个生命周期内进行;控制和监视测试过程
管理和度量级	测试是一个度量和质量控制的过程。在软件生命周期中评审被作为测试和软件质量控制的一部分	进行可靠性、可用性和可维护性等方面的测试;采用数据库来管理测试用例;具有缺陷管理系统并划分缺陷的级别;还没有建立起缺陷预防机制,缺乏自动对测试中产生的数据进行收集和分析的手段	实施软件生命周期中各阶段评审;建立测试数据库并记录、收集有关的测试数据;建立组织范围内的评审程序;建立测试过程的度量方法和程序;进行软件质量评价
优化级	具有缺陷预防和质量控制的能力,已经建立起测试规范和流程,并不断地进行测试改进	运用缺陷预防和质量控制措施;选择和评估测试工具存在一个既定的流程;测试自动化程度高;自动收集缺陷信息;有常规的缺陷分析机制	应用过程数据预防缺陷,统计质量控制,建立软件产品的质量目标,持续改进、优化测试过程

2. 软件测试工具分类

1) 白盒测试工具

(1) 静态测试工具。

职能:主要集中在需求文档、设计文档以及程序结构上,可以进行类型分析、接口分析、输入输出规格说明分析等。

工具:McCabe & Associates公司开发的McCabe Visual Quality ToolSet分析工具,ViewLog公司开发的LogiScope分析工具,Software Research公司开发的TestWork/Advisor分析工具,Software Emancipation公司开发的Discover分析工具,北

京邮电大学开发的 DTS 缺陷测试工具等。

(2) 动态测试工具。

职能：功能确认与接口测试、覆盖率分析、性能分析、内存分析等。

工具：Compuware 公司开发的 DevPartner 软件，Rational 公司研制的 Purify 系列等。

2) 黑盒测试工具

工具：Rational 公司的 TeamTest，Compuware 公司的 QACenter。

分类：功能测试工具，性能测试工具。

3) 测试设计和开发工具

职能：测试设计是说明测试被测软件特征或特征组合的方法，确定并选择相关测试用例的过程。

测试开发是将测试设计转换成具体的测试用例的过程。

工具：测试数据生成器，基于需求的测试设计工具，捕获/回放工具和覆盖分析工具。

4) 测试执行和评估工具

职能：执行测试用例并对结果进行评估，包括选择用于执行的测试用例，设置测试环境，运行所选择的测试，记录测试执行活动，分析潜在的软件故障并测量测试工作的有效性。

工具：捕获/回放，覆盖分析；存储器测试。

5) 测试管理工具

职能：帮助测试人员完成测试计划，跟踪测试运行结果等，主要用于测试用例管理、缺陷跟踪管理、配置管理。

工具：Rational 公司的 Test Manager，Compuware 公司的 TrackRecord 等。

7.3 典型习题解析

7.3.1 选择题

1. 创建一个基于 JUnit 的单元测试类，该类必须继承(　　)类。

A. TestSuite　　B. Assert　　C. TestCase　　D. JFCTestCase

【答案】 C

2. 以下关于单元测试不正确的说法是(　　)。

A. 单元测试的主要目的是针对编码过程中可能存在的各种错误

B. 单元测试一般是由程序开发人员完成的

C. 单元测试是一种不需要关注程序结构的测试

D. 单元测试属于白盒测试的一种

【答案】 C

3. 测试驱动开发的含义是(　　)。

A. 先写程序后写测试的开发方法　　B. 先写测试后写程序，即“测试先行”

C. 用单元测试的方法写测试　　D. 不需要测试的开发

【答案】 B

4. 用 JUnit 断言一个方法输出的是指定字符串,应当用的断言方法是(　　)。

A. assertNotNull()　　B. assertSame()

C. assertEquals()　　D. assertNotEquals()

【答案】 C

5. TestCase 是 junit.framework 中的一个(　　)。

A. 方法　　B. 接口　　C. 类　　D. 抽象类

【答案】 C

6. TestSuite 在 JUnit 中的作用是(　　)。

A. 集成多个测试用例　　B. 做系统测试用的

C. 做自动化测试用的　　D. 方法断言

【答案】 A

7. 对于测试程序的命名规则,以下说法正确的是(　　)。

A. 测试类的命名只要符合 Java 类的命名规则就可以了

B. 测试类的命名一般要求以 Test 打头,后接类名称,如 TestPerson

C. 测试类的命名一般要求以 Test 结尾,前面加上类名称,如 PersonTest

D. 测试类中的方法都是以 testXxx()形式出现

【答案】 C

8. 初始化一个被测试对象通常会在测试类的(　　)中进行。

A. teardown(　　)　　B. setup(　　)

C. 构造方法　　D. 任意位置

【答案】 B

9. 以下不属于单元测试优点的是(　　)。

A. 单元测试是一种验证行为　　B. 单元测试是一种设计行为

C. 单元测试是一种编写文档的行为　　D. 单元测试是一种评估行为

【答案】 D

10. 从技术角度划分的测试类型不包括(　　)。

A. 黑盒测试　　B. 白盒测试　　C. 单元测试　　D. 灰盒测试

【答案】 C

11. 以下关于 JUnit 的特征的叙述不正确的是(　　)。

A. 用于测试期望结果的断言

B. 用于共享共同测试数据的测试工具

C. 易于集成到测试人员的构建过程中,JUnit 和 Ant 的结合可以实施增量开发

D. JUnit 是收费的,不能做二次开发

【答案】 D

12. JUnit 的两个模式是集成模式和(　　)。

A. 命令模式　　B. 适配器模式　　C. 单例模式　　D. 接口模式

【答案】 A

13. 测试 6 的阶乘,断言方法是(　　)。

A. Assert.assertSame(720, jc. jieChen(6))

B. Assert.assertEquals(720, jc. jieChen(6))

C. Assert.assertNull(720, jc. jieChen(6))

D. Assert.assertTrue(720, jc. jieChen(6))

【答案】 B

7.3.2 简答题

1. TDD 是什么?

【答】 TDD 的意思是测试驱动开发,它是一种以"测试先行"为原则的开发方法,开发人员在编写产品代码前,通常先写出对应的测试程序,然后再编写产品代码,最后再进行测试。

2. 自动化测试工具有哪些特征?

【答】 自动化测试工具有以下特征:

(1) 支持脚本语言的函数库。

支持脚本语言的函数库是对测试工具最基本的要求。程序即使作了修改,只需要修改原脚本中的相应函数,而不用改动所有涉及的脚本,节省大量工作。另外,通过对外部函数的支持,如对 DLL 文件的访问,对数据库编程接口的调用,获得强大的功能。

(2) 对程序界面中对象的识别能力。

测试工具必须能够区分和识别程序界面中相应的对象(如按钮、文本框、表单等),录制的测试脚本才能具有良好的可读性、修改的灵活性和维护的方便性。如果只是简单地通过像素位置坐标区分对象,就会存在较多的问题,例如界面稍微改变或者屏幕的分辨率、测试环境发生改变,会导致原有的测试脚本无法使用。

(3) 抽象层。

抽象层和对象识别能力有一定的关系。在捕捉回放程序界面过程中,抽象层一般位于被测应用程序和录制生成的测试脚本之间,抽象层用于将程序界面中存在的对象实体映射成逻辑对象,测试针对逻辑对象进行,不依赖于界面的对象实体,能够减少测试脚本建立和维护的工作量。

(4) 分布式测试的网络支持。

互联网软件,如网络会议系统、远程培训系统、聊天系统等软件,一般都具有协同工作、相互通信等模式,支持多用户共同操作,这些软件的测试有如下要求:

① 测试工具在进行测试时传输的数据量要小,具有独立性,避免影响被测软件。

② 按照事先设置的任务执行时间表进行,在指定时间、指定设备上执行指定测试任务。

③ 当两个测试任务并发时,应能保持协调或协同处理,避免出现资源竞争问题。

(5)图表功能。

测试工具能够将测试结果生成一些统计报表,利于测试人员的工作。

(6)测试工具的集成能力。

测试工具伴随着测试过程的改进是一个持续的过程。因此,测试工具应该和开发工具良好地集成。

3. 什么是自动化测试?

【答】 自动化测试是指"一切可以由计算机系统自动完成的测试任务都已经由计算机系统或软件工具、程序来承担并自动执行",自动化测试具有执行速度快、测试结果准确、复用性高、可靠等特点。

4. 软件测试工具如何进行分类?

【答】 软件测试工具可以分为以下5类:

(1)负载压力测试工具。通过模拟成百上千个用户并发执行业务操作来完成对应用程序的测试。主要用于度量应用系统的可扩展性和性能,并通过实时性能监测来确认和查找问题。

(2)功能测试工具。通过自动录制、检测和回放用户的应用操作,将被测系统的输出记录与预先设定的标准结果自动进行比较,以检测应用程序是否能够达到预期功能并正常运行。此类工具可以大大减少黑盒测试的工作量,并能很好地进行回归测试。

(3)单元测试工具。通过自动执行应用程序的函数、过程或完成某个特定功能的程序块,将程序运行结果与预先设置的标准结果自动进行比较,以检测函数、过程或功能是否达到预期结果。与功能测试工具的最大不同之处在于此类工具属于白盒测试工具,且通常由开发人员自行完成。

(4)代码质量测试工具。根据预定义的语法规则对代码进行扫描,找出不符合编码规范的地方。

(5)测试管理工具。用于对测试需求、测试计划、测试用例、测试实施进行管理,将测试过程流水化,让不同人员可以通过工具实时交换相关信息,实现全过程的自动化管理。

5. 静态分析工具有哪几类功能?

【答】

(1)对模块中的所有变量,检查其是否都已定义,是否引用了未定义的变量,是否有已赋过值但从未使用的变量。实现方法是建立变量的交叉引用表。

(2)检查模块接口的一致性。

(3)检查在逻辑上可能有错误的结构以及多余的不可达的程序段。

(4)建立"变量/语句交叉引用表""子程序调用顺序表""公共区/子程序交叉引用表"等。利用它们找出变量错误可能影响到哪些语句,影响到哪些其他变量等。

(5)检查被测程序违反编程标准的错误,例如模块大小、模块结构、注释的约定、某些语句形式的使用以及文档编制的约定等。

(6) 对一些静态特性的统计功能。对于各种类型源程序的出现次数,标识符使用的交叉索引,标识符在每个语句中使用的情况,函数与过程引用情况,任何输入数据都执行不到的孤立代码段,未经定义的或未曾使用过的变量,违背编码标准之处,公共变量与局部变量的各种统计。

6. 自动化测试的优点有哪些?

【答】

(1) 回归测试更方便。特别是在程序修改比较频繁时,效果是非常明显的。由于回归测试的动作和用例是完全设计好的,测试期望的结果也是完全可以预料的,自动运行回归测试,可以极大地提高测试效率,缩短回归测试时间。

(2) 运行更多更烦琐的测试,在较少的时间内能运行更多的测试。

(3) 执行一些手工测试困难或不可能进行的测试。比如,对于大量用户的测试,不可能同时让足够多的测试人员同时进行测试,但是可以通过自动化测试模拟同时有许多用户的情况,从而达到测试的目的。

(4) 测试的复用性。由于自动测试通常采用脚本技术,这样就有可能只需要做少量修改甚至不做修改,就能实现在不同的测试过程中使用相同的用例。

7. 如何理解软件测试自动化?

【答】 软件测试自动化是一项让计算机代替测试人员进行软件测试的技术,可以让测试人员从烦琐和重复的测试活动中解脱出来,专心从事有意义的测试设计等活动。如果采用自动比较技术,还可以自动完成测试用例执行结果的判断,从而避免人工比对存在的疏漏问题。设计良好的自动化测试,在某些情况下可以实现"夜间测试"和"无人测试"。在大多数情况下,软件测试自动化可以减少开支,增加有限时间内可执行的测试数量,在执行相同数量的测试时节约测试时间。

软件测试自动化通常借助测试工具进行。测试工具可以进行一部分测试设计、实现、执行和比较的工作。通过运用测试工具,可以达到提高测试效率的目的,所以对测试工具的选择和推广使用应该给予重视。部分测试工具可以实现测试用例的自动生成,但通常的工作方式为人工设计测试用例,使用工具进行用例的执行和比较。

软件测试自动化的设计并不能由工具来完成,必须由测试人员进行手工设计,但是在设计时必须考虑自动化测试的特殊要求,否则无法实现利用工具进行测试用例的自动执行。为此,就必须在测试的设计和内容的组织方面采取一些特殊的方法。

对于软件测试自动化的工作,大多数人都认为是一件非常容易的事。其实,软件测试自动化的工作量非常大,而且也并不是在任何情况下都适用,同时软件测试自动化的设计并不比程序设计简单。

8. 自动化测试的发展经过了几个阶段?

【答】 如图 7.1 所示,自动化测试的发展大致经历了如下 4 个阶段:

第 1 阶段,机械方式实现人工重复操作。最初,自动化测试主要研究如何采用自动方

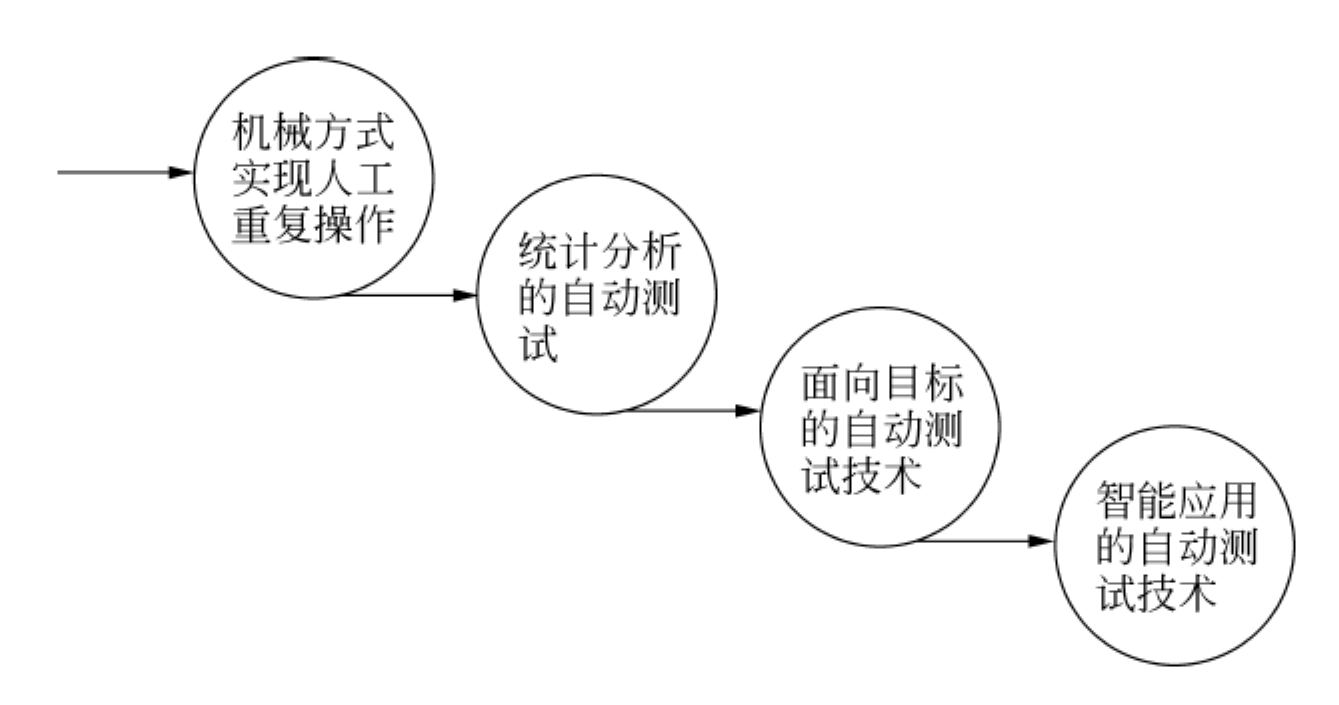


图 7.1 自动化测试发展阶段

法来实现和替代人工测试中的烦琐和机械重复的工作，主要包括自动生成测试数据，对程序动态执行检测，脚本驱动自动执行。此时的自动测试活动只是软件测试过程的偶然行为，在一定程度上提高了效率，简化了测试人员的工作，但对整体的测试过程并无太大改进。

第 2 阶段，统计分析的自动测试。只有自动测试结果具有可靠性，其使用才具有实际的意义。在这一阶段，针对不同的测试准则和测试策略，指导测试的自动化过程以及对测试的结果进行评估。

第 3 阶段，面向目标的自动测试技术。软件测试并不是机械和随机地发现错误，而是带有很强的目的性。在这一阶段，进化计算和人工智能等技术，以及各种高性能的算法被引入自动测试技术。

第 4 阶段，智能应用的自动测试技术。在这一阶段，能力成熟度模型被引入软件工程，测试业也出现了对应的测试成熟度模型。不同的自动测试等级成为测试好坏的一个衡量依据。

9. 测试成熟度模型是什么？其包括哪些内容？

【答】 测试成熟度模型描述了测试过程，使得项目测试部分有了良好的计划和控制的基础。测试成熟度分为如下 5 个级别：初始级、定义级、集成级、管理和度量级、优化级。

TMM 5 个级别可概括地描述如下：

第一级：测试和调试没有区别，除了支持调试外，测试没有其他目的。

第二级：测试的目的是为了表明软件能够工作。

第三级：测试的目的是为了表明软件能够正常工作。

第四级：测试的目的是为了把软件不能正常工作的预知风险降低到能够接受的程度。

第五级：测试成为自觉的约束。

10. 测试框架是什么？当前有几代测试框架？

【答】 测试框架是为自动化软件测试提供支持的集合。测试框架经历了如下发展

过程：

第一代，无测试框架。测试需求与测试用例关联非常弱，对自动化测试人员的编程水平以及测试方面的整体知识要求都非常高。

第二代，部分的测试框架。完成部分自动化测试管理，实现对于测试用例管理和缺陷跟踪，面向业务流。实现自动化测试需编写程序。

第三代，完整的测试框架。具有如下功能：

(1) 拥有数据场景管理。

(2) 企业级的面向工作流的缺陷管理。

(3) 业务流复用框架。

(4) 高伸缩的自动执行框架。

11. 录制和回放是指什么？

【答】 目前的自动化负载测试解决方案几乎都是采用“录制-回放”的技术。录制是通过捕获用户每一步操作及结果，如用户界面的像素坐标或程序显示对象(窗口、按钮、滚动条等)的位置以及相应的状态变化或属性变化，用一种脚本语言记录下来，模拟用户操作。回放是将脚本语言转换为屏幕操作，比较被测系统的输出记录与预先给定的标准结果。

7.3.3 设计题

1. 采用 JUnit 单元测试框架编写一个 N 的阶乘程序，并对其进行测试。

N 的阶乘程序代码如下：

```
import java.math.BigInteger;
import java.util.Scanner;
public class JieChen {
    long result=1;
    long doJieChen(int a){
        for(int i=2; i<=a; i++){
            result*=i;
        }
        return result;
    }
    public static void main(String args []){
        JieChen jc=new JieChen();
        Scanner c=new Scanner(System.in);
        System.out.print(jc.doJieChen(c.nextInt()));
    }
}
```

【解析】

采用 JUnit 测试 N 的阶乘程序的测试代码如下：

```
import junit.framework.TestCase;
public class JieChenTest extends TestCase {
    /* (non-Javadoc)
     * @see junit.framework.TestCase#setUp()
     */
    JieChen j;
    protected void setUp()throws Exception {
        super.setUp();
        j=new JieChen();
    }
    public void testDoJieChen(){
        assertEquals("这是测试阶乘的值:",24,j.doJieChen(4));
    }
}
```

2. 实现将字符串全部变成小写字母的转换器,并对其进行测试。

要求:有正确、完整的程序代码和测试代码,基于JUnit单元测试框架进行测试。

【解析】

实现将字符串全部变成小写字母的Java代码如下:

```
package demo;
public class StringDemo {
    public StringDemo(){
    }
    public String smallString(String str)                //字符串变小写
    {
        String temp="error";
        if(str.equals("")|| str ==null){
            return temp;
        }
        String s=str.toLowerCase();
        return s;
    }
}
```

采用JUnit进行测试的测试代码如下:

```
package demo.test;
import demo.StringDemo;
import junit.framework.TestCase;
public class testStringDemo extends TestCase {
    private StringDemo str;
    protected void setUp(){
        str=new StringDemo();
    }
```

```
    public void testSmallString(){
        assertEquals("一个字母变小写",str.smallString("A"),"a");
        assertEquals("字符串全是大写",str.smallString("ABC"), "abc");
        assertEquals("含有小写的字符串",str.smallString("aBc"), "abc");
        assertEquals("含有数字的字符串",str.smallString("A1C"), "a1c");
        assertEquals("全是数字的字符串",str.smallString("123"), "123");
        assertEquals("含有特殊字符的处理",str.smallString(",Adc"), ",adc");
        assertEquals("异常处理",str.smallString(""),"error");
    }
}
```

3. 采用Java语言模拟日历并且可以计算选定日期的下一天的日期的程序,包括CalendarUnit类、Date类、NextDay类、Year类、Month类和Day类。

CalendarUnit类代码如下:

```
public abstract class CalendarUnit {
    protected int currentPos;
    protected void setCurrentPos(int pCurrentPos){
        currentPos=pCurrentPos;
    }
    protected int getCurrentPos(){
        return currentPos;
    }
    protected abstract boolean increment();
    protected abstract boolean isValid();
}
```

Date类代码如下:

```
public class Date {
    private Day d;
    private Month m;
    private Year y;
    public Date(int pMonth, int pDay, int pYear){
        y=new Year(pYear);
        m=new Month(pMonth, y);
        d=new Day(pDay, m);
    }
    public void increment(){
        if(!d.increment()){
            if(!m.increment()){
                y.increment();
                m.setMonth(1, y);
            }
            d.setDay(1, m);
        }
```

```
    }
    public void printDate(){
        System.out.println(m.getMonth()+"/"+d.getDay()+"/"+y.getYear());
    }
    public Day getDay(){
        return d;
    }
    public Month getMonth(){
        return m;
    }
    public Year getYear(){
        return y;
    }
    public boolean equals(Object o){
        if(o instanceof Date){
            if(this.y.equals(((Date)o).y) && this.m.equals(((Date)o).m) && this.d.equals
            (((Date)o).d))
                return true;
        }
        return false;
    }
    public String toString(){
        return(m.getMonth()+"/"+d.getDay()+"/"+y.getYear());
    }
}
```

Nextday 类代码如下：

```
public class Nextday {
    public static Date nextDay(Date d){
        Date dd= new Date(d.getMonth().getCurrentPos(), d.getDay().getCurrentPos(), d.
        getYear().getCurrentPos());
        dd.increment();
        return dd;
    }
}
```

Year 类代码如下：

```
public class Year extends CalendarUnit {
    public Year(int pYear){
        setYear(pYear);
    }
    public void setYear(int pYear){
        setCurrentPos(pYear);
        if(!this.isValid()){
```

```
                throw new IllegalArgumentException("Not a valid month");
            }
        }
        public int getYear(){
            return currentPos;
        }
        public boolean increment(){
            currentPos=currentPos+1;
            if(currentPos ==0)
                currentPos=1;
            return true;
        }
        public boolean isLeap(){
            if(currentPos >=0 &&(((currentPos %4 ==0)&&(currentPos %100 !=0))||(currentPos
            %400 ==0)))
                return true;
            else if(currentPos<0 &&((((currentPos * -1)%4 ==1)&&
            ((currentPos * -1)%100 !=1))||((currentPos * -1)%400 ==1)))
                return true;
            return false;
        }
        protected boolean isValid(){
            if(this.currentPos !=0)
                return true;
            return false;
        }
        public boolean equals(Object o){
            if(o instanceof Year){
                if(this.currentPos ==((Year)o).currentPos)
                    return true;
            }
            return false;
        }
    }
```

Month 类代码如下：

```
public class Month extends CalendarUnit {
    private Year y;
    private int[] sizeIndex={ 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
    public Month(int pMonth, Year y){
        setMonth(pMonth, y);
    }
    public void setMonth(int pMonth, Year y){
        setCurrentPos(pMonth);
```

```
            this.y=y;
            if(!this.isValid()){
                throw new IllegalArgumentException("Not a valid month");
            }
        }
        public int getMonth(){
            return currentPos;
        }
        public int getMonthSize(){
            if(y.isLeap())
                sizeIndex[1]=29;
            else
                sizeIndex[1]=28;
            return sizeIndex[currentPos-1];
        }
        public boolean increment(){
            currentPos +=1;
            if(currentPos>12)
                return false;
            else
                return true;
        }
        public boolean isValid(){
            if(y !=null && y.isValid())
                if(this.currentPos >=1 && this.currentPos <=12)
                    return true;
            return false;
        }
        public boolean equals(Object o){
            if(o instanceof Month){
                if(this.currentPos == ((Month)o).currentPos
                        && this.y.equals(((Month)o).y))
                    return true;
            }
            return false;
        }
    }
```

Day 类代码如下：

```
public class Day extends CalendarUnit {
    private Month m;
    public Day(int pDay, Month m){
        setDay(pDay, m);
    }
```

```
    public boolean increment(){
        currentPos +=1;
        if(currentPos <=m.getMonthSize())
            return true;
        else
            return false;
    }
    public void setDay(int pDay, Month m){
        setCurrentPos(pDay);
        this.m=m;
        if(!this.isValid()){
            throw new IllegalArgumentException("Not a valid day");
        }
    }
    public int getDay(){
        return currentPos;
    }
    public boolean isValid(){
        if(m !=null && m.isValid())
            if(this.currentPos >=1 && this.currentPos <=m.getMonthSize())
                return true;
        return false;
    }
    public boolean equals(Object o){
        if(o instanceof Day){
            if(this.currentPos ==((Day)o).currentPos &&
                    this.m.equals(((Day)o).m))
                return true;
        }
        return false;
    }
}
```

要求采用 JUnit 单元测试框架进行测试。

【解析】

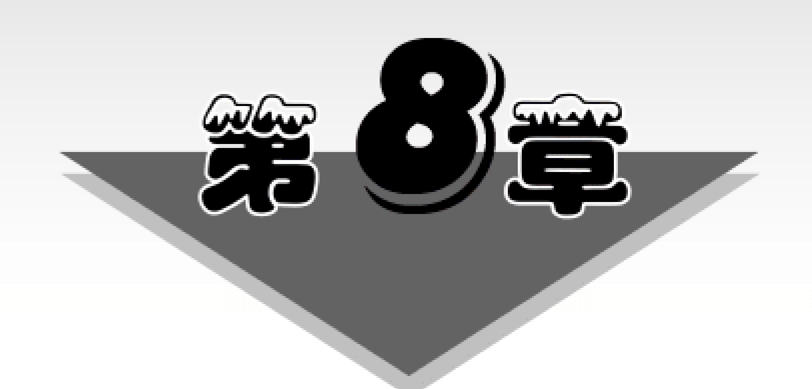
测试代码如下：

```
import static org.junit.Assert.* ;
import javax.security.auth.login.FailedLoginException;
import org.junit.Before;
import org.junit.Test;
public class TestNextDay {
    Nextday nextday;
    @Before
    public void setUp(){
```

```
        nextday=new Nextday();
    }
    @Test
    public void testYear(){
        Year year;
        try{
            year=new Year(0);
            fail("There should be an exceptiion");
        }
        catch(Exception expected){
            assertTrue(true);
        }
        year =new Year(2000);
        assertEquals(2000,year.getYear());
        //isLeap
        assertEquals(true, year.isLeap());
        year=new Year(-1997);
        assertTrue(year.isLeap());
        year=new Year(1995);
        assertFalse(year.isLeap());
        //increment
        year.currentPos=-1;
        year.increment();
        assertEquals(1,year.getYear());
        year.increment();
        assertEquals(2, year.getYear());
        //equals
        Month month=new Month(1,year);
        assertFalse(year.equals(month));
        Year year2=new Year(2);
        assertTrue(year.equals(year2));
        year2=new Year(-1996);
        assertFalse(year.equals(year2));
    }
    @Test
    public void testMonth(){
        Month month;
        Year year=null;
        try{
            month=new Month(1, year);
            fail("There should be an exception");
        }
        catch(Exception e){
            assertTrue(true);
```

```
        }
        year=new Year(1);
        try{
            month=new Month(0, year);
            fail("There should be an exception");
        }
        catch(Exception e){
            assertTrue(true);
        }
        month=new Month(2,year);
        year=new Year(1996);
        Month month2=new Month(2, year);
        assertFalse(month.equals(month2));
        //monthsize
        assertEquals(28,month.getMonthSize());
        assertEquals(29,month2.getMonthSize());
        //increment
        month=new Month(12,year);
        assertFalse(month.increment());
        month=new Month(1,year);
        assertEquals(true,month.increment());
        //equals
        assertFalse(month.equals(year));
        month2=new Month(2,year);
        assertTrue(month.equals(month2));
    }
    @Test
    public void testDay(){
        Year year;
        Month month;
        Day day;
        //setDay
        year=new Year(1996);
        month=new Month(2, year);
        try{
            day=new Day(30,month);
            fail("There should be an exception");
        }
        catch(Exception e){
            assertTrue(true);
        }
        month=null;
        try{
            day=new Day(30,month);
```

```
            fail("There should be an exception");
        }
        catch(Exception e){
            assertTrue(true);
        }
        month=new Month(2, year);
        day=new Day(29,month);
        assertTrue(day.isValid());
        //increment
        assertFalse(day.increment());
        day=new Day(27, month);
        assertTrue(day.increment());
        //equals
        assertFalse(day.equals(month));
        Day day2=new Day(27,month);
        assertFalse(day.equals(day2));
        day2.increment();
        assertTrue(day.equals(day2));
    }
    @Test
    public void testDate(){
        Date date,date2;
        //increment
        date=new Date(1, 1, 1999);
        date2=new Date(1, 2, 1999);
        date.increment();
        assertEquals(date, date2);
        date=new Date(2, 29, 1996);
        date2=new Date(3, 1, 1996);
        date.increment();
        assertEquals(date, date2);
        date=new Date(12, 31, 1996);
        date2=new Date(1,1, 1997);
        date.increment();
        assertEquals(date, date2);
        //euqals
        assertTrue(date.equals(date2));
        assertFalse(date.equals(date2.getDay()));
        date2.increment();
        assertFalse(date.equals(date2));
    }
}
```



软件测试管理

8.1 本章要求

- 了解测试管理体系。
- 理解测试过程改进。
- 理解测试文档管理。
- 理解人力资源。
- 理解配置管理。
- 了解软件质量的内容。

8.2 本章知识重点

1. 测试管理体系

测试系统主要由测试计划、测试设计、测试实施、配置管理、资源管理、测试管理 6 个相互关联、相互作用的过程组成。

2. 测试组织管理

测试组织管理主要包括以下工作：

- 组织和管理测试小组。
- 确定测试小组的组织模式。
- 安排测试任务。
- 估计测试工作量。
- 确定应交付的测试文档。
- 管理测试软件。
- 确定测试需求和组织测试设计等。

3. 软件质量与软件测试相关的特性

软件质量是一个复杂的多层面概念。

(1) 从用户角度出发，质量是对需求的满足。软件需求是度量软件质量的基础。

(2) 从软件产品角度出发,质量是软件的内在特征。

(3) 从软件开发过程出发,质量是对过程规范的符合。

4. 软件测试文档

测试文档大致分为测试计划和测试分析报告。

(1) 测试计划:详细规定测试的要求,包括测试的目的、内容、方法、步骤以及测试的准则等。

(2) 测试分析报告:用来对测试结果进行分析说明。软件经过测试后,应给出评价的结论性意见,如软件的能力如何,存在哪些缺陷和限制等。

5. 人力资源

图 8.1 给出了测试过程组织的框架图。

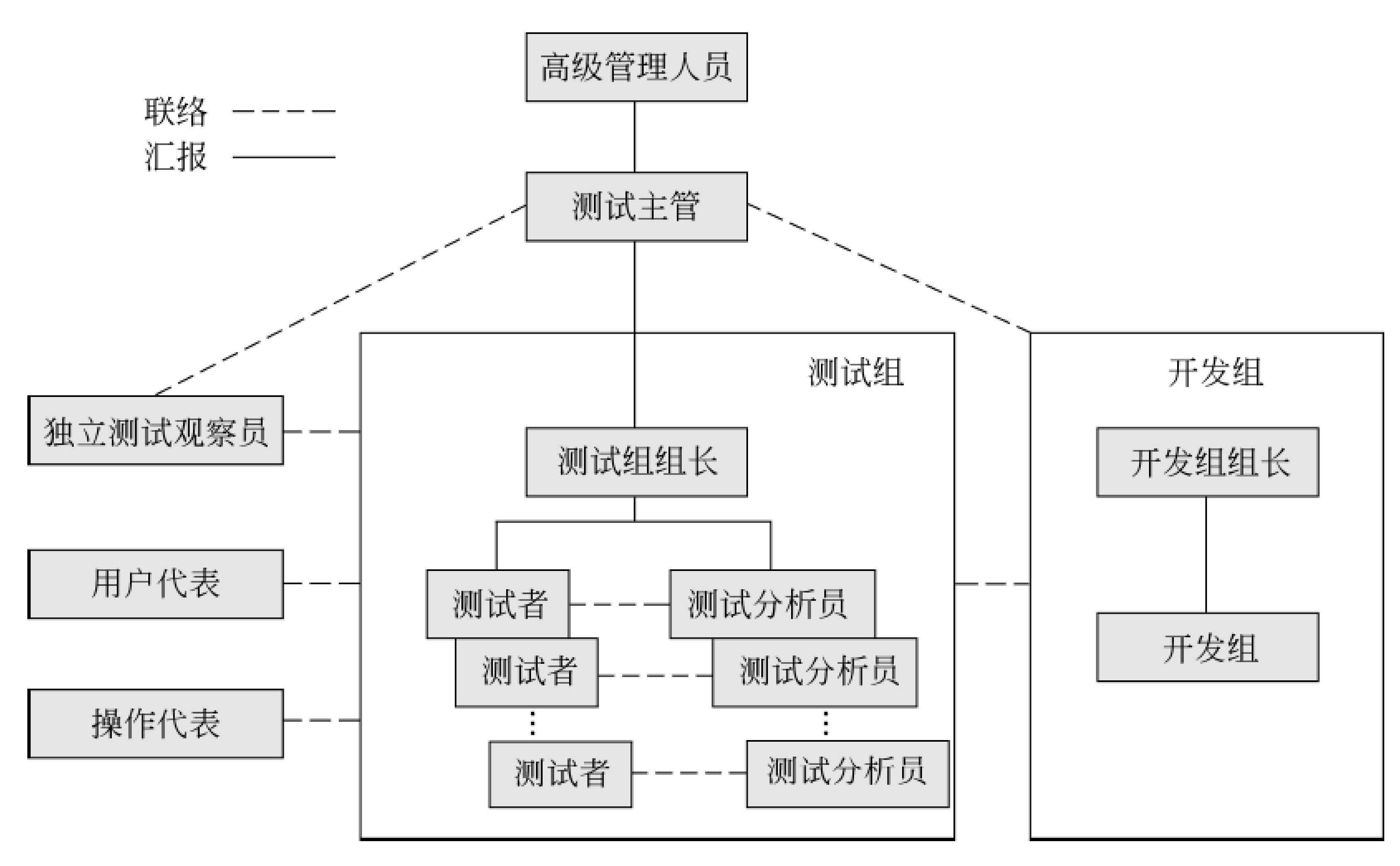


图 8.1 测试过程组织

6. 测试管理工具

测试管理工具是指利用工具对测试输入、执行和结果进行管理。一般而言,测试管理工具对测试计划、测试用例、测试实施进行管理,还包括对缺陷的跟踪管理。测试管理工具能提高测试效率、测试质量、测试用例复用率等。目前市场上主流的软件测试管理工具有 TestCenter、TestDirector、TestLink 等。

8.3 典型习题解析

8.3.1 选择题

1. 为保证测试活动的可控性,必须在软件测试过程中进行软件测试配置管理,一般

来说，软件测试配置管理中最基本的活动包括(　　)。

A. 配置项标识、配置项控制、配置报告状态、配置审计

B. 配置基线确立、配置项控制、配置报告、配置审计

C. 配置项标识、配置项变更、配置审计、配置跟踪

D. 配置项标识、配置项控制、配置项报告状态、配置跟踪

【答】 A

2. 风险管理已经成为软件工程项目管理的一项重要内容，其主要活动包括(　　)。

A. 风险定义、风险估计、风险应对、风险控制

B. 风险识别、风险估计、风险应对、风险控制

C. 风险定义、风险分解、风险应对、风险控制

D. 风险识别、风险分解、风险应对、风险控制

【答】 B

3. 测试设计员的职责有(　　)。

A. 制定测试计划　　B. 设计测试用例

C. 设计测试过程、脚本　　D. 评估测试活动

【答】 BC

4. 软件文档不仅是软件开发各阶段的重要依据，而且影响软件的(　　)。

A. 可理解性　　B. 可维护性　　C. 可扩展性　　D. 可移植性

【答】 D

5. 测试设计员的职责有(　　)。

A. 制定测试计划　　B. 设计测试用例

C. 设计测试过程、脚本　　D. 评估测试活动

【答】 BC

6. 对测试文档的要求是(　　)。

A. 为以后的跟踪提供依据　　B. 能证实测试过程

C. 能证实测试步骤，要覆盖开发生命周期　　D. 以上都对

【答】 D

7. GB/T 9386《计算机软件测试文件编辑指南》详细描述了计算机软件测试文件应该包含的内容及编写格式，并将测试文件分为(　　)两类。

A. 测试计划和测试过程细则　　B. 测试计划和测试分析报告

C. 测试过程定义和测试分析报告　　D. 测试数据和测试分析报告

【答】 B

8. 软件测试是软件质量保证的重要手段，下列(　　)是软件测试的任务。

①预防软件发生错误；②发现改正程序错误；③提供诊断错误信息。

A. ①　　B. ②　　C. ③　　D. 以上都对

【答】 D

9. 软件的六大质量特性包括(　　)。

A. 功能性、可靠性、可用性、效率、可维护、可移植

B. 功能性、可靠性、可用性、效率、稳定性、可移植

C. 功能性、可靠性、可扩展性、效率、稳定性、可移植

D. 功能性、可靠性、兼容性、效率、稳定性、可移植

【答案】 A

10. 软件测试是软件质量保证的主要手段之一，测试的费用已超过(　　)的30%以上。

A. 软件开发费用　　B. 软件维护费用

C. 软件开发和维护费用　　D. 软件研制费用

【答案】 A

11. 软件质量在软件测试中被定义为(　　)。

A. 正确程度　　B. 适于使用或符合要求

C. 人们对软件需求的程度　　D. 软件的用途和适用范围

【答案】 C

12. 软件质量管理由质量保证和质量控制组成，下面的选项中(　　)属于质量控制。

A. 测试　　B. 跟踪　　C. 监督　　D. 制定计划

E. 需求审查　　F. 程序代码审查

【答案】 ABC

13. 进行软件质量管理的重要性有(　　)。

A. 降低维护成本　　B. 法律上的要求

C. 市场竞争的需要　　D. 质量标准化的趋势

E. 软件工程的需要　　F. CMM过程的一部分

G. 方便与客户进一步沟通，为后期的实施打好基础

【答案】 ABCD

14. 测试设计员的职责有(　　)。

A. 制定测试计划　　B. 设计测试用例

C. 设计测试过程、脚本　　D. 评估测试活动

【答案】 BC

15. 软件测试计划评审会需要(　　)参加。

A. 项目经理　　B. 软件质量保证负责人

C. 配置负责人　　D. 测试组

【答案】 ABCD

8.3.2 简答题

1. 测试项目具有哪些特点？

【答】 软件测试项目具有如下特点：

(1) 软件测试项目的各个里程碑标准和度量的定义、管理要求更高。

(2) 软件测试项目的变化控制更高。随着系统分析、设计和实施的进行，用户需求变

化导致项目进度、程序代码及相关文档的一系列变化，软件测试也应随之变化。

(3) 软件测试项目具有智力密集型特点，受人力影响较大，项目团队中人的能力以及团队的结构和稳定性对测试执行、产品质量影响很大。

(4) 测试任务分配难。例如，单元测试、集成测试、系统测试和验收测试等关联比较大，但要求的技术往往不同，不容易分割进行单独测试。

总之，软件测试项目管理包括质量管理、人力资源管理、沟通管理、风险管理等，对软件质量具有直接影响。

2. CMM是什么？具体内容是什么？

【答】 CMM(Capability Maturity Model)是能力成熟度模型的缩写。作为软件质量的保证措施，CMM 为软件企业的过程能力提供了一个阶梯式的进化框架，该框架共有5级，分别是初始级、可重复、已定义级、已管理级和优化级。它和 CMMI 模型对加强软件过程管理、软件项目管理、软件组织自身的建设起到了重要推动作用，同时也为客户选择软件开发组织提供了科学的依据。

CMM 包括如下两部分：软件能力成熟度模型和能力成熟度模型的关键惯例。软件能力成熟度模型主要是描述模型的结构，给出模型的基本构件的定义。能力成熟度模型的关键惯例详细描述了每个关键过程方面涉及的关键惯例。

3. 简述软件测试管理的内容。

【答】 软件测试管理认为，软件测试是一个复杂的系统工程，需要对组成这个系统的各个部分进行识别和管理，实现特定的系统目标。测试系统主要由测试计划、测试设计、测试实施、配置管理、资源管理、测试管理 6 个过程组成。

软件测试管理体系一般包括如下 6 个步骤：

(1) 识别软件测试所需的过程及其应用，即测试计划、测试设计、测试实施、配置管理、资源管理、测试管理。

(2) 确定这些过程的顺序和相互作用，前一个过程的输出作为后一个过程的输入。其中，配置管理和资源管理是支撑性的过程。

(3) 确定这些过程所需要的准则和方法，制定这 6 个过程所需的文档。

(4) 确保所需的资源和信息，并对这 6 个过程进行监测。

(5) 监视、测量和分析这些过程。

(6) 实施必要的过程改进措施。

4. 影响软件质量的主要因素有哪些？

【答】 从管理角度对软件质量进行度量，影响软件质量的主要因素可以分成以下几类：

(1) 正确性。系统满足规格说明和用户目标的程度，即在预定环境下能正确地完成预期功能的程度。

(2) 健壮性。在系统发生故障、输入的数据无效或操作错误等情况下，系统能做出适

当响应的程度。

(3) 效率。为了完成预定的功能,系统需要计算资源的多少。

(4) 安全性。对未经授权的人使用软件或数据的企图,系统能够控制(禁止)的程度。

(5) 可用性。系统在完成预定功能时令人满意的程度。

(6) 风险。按预定的成本和进度把系统开发出来,并且让用户满意的概率。

(7) 可理解性。理解和使用该系统的容易程度。

(8) 可维护性。诊断和改正在运行现场发现的错误所需要的工作量。

(9) 适应性。修改或改进正在运行的系统所需要的工作量。

(10) 可测试性。软件容易测试的程度。

(11) 可移植性。把程序从一种硬件配置和软件系统环境转移到另一种配置和环境时需要的工作量。

(12) 可再用性。在其他应用中该程序可以被再次使用的程度(或范围)。

(13) 互运行性。把该系统和另一个系统结合起来的工作量。

5. 为了在软件开发过程中保证软件的质量,应主要采取哪些措施?

【答】 为了在软件开发过程中保证软件的质量,应主要采取下述措施。

(1) 审查。在软件生命周期每个阶段结束之前,使用结束标准对该阶段生产出的软件配置成分进行严格的技术审查。审查小组通常由 4 人组成:组长、作者和两名评审员。组长负责组织和领导技术审查,作者是开发文档或程序的人,两名评审员由与评审结果利害攸关的人担任,负责提出技术评论。

(2) 复查和管理复审。复查即是检查已有的材料,以断定某阶段的工作是否能够开始或继续。每个阶段开始时的复查是为了肯定前一个阶段结束时确实进行了认真的审查,已经具备了开始当前阶段工作所必需的材料。管理复审通常是指向开发组织或使用部门的管理人员提供有关项目的总体状况、成本和进度等方面的情况,以便他们从管理角度对开发工作进行审查。

(3) 测试。用已知的输入在已知环境中动态地执行系统或系统的部件。如果测试结果与预期结果不一致,则表明系统中可能出现了错误。

6. 软件质量属性有哪些?

【答】 软件质量属性划分为开发期质量属性和运行期质量属性两大类。

(1) 开发期质量属性。包含和软件开发、维护和移植这 3 类活动相关的所有质量属性,这些是开发人员、开发管理人员和维护人员都非常关心的,对最终用户而言,这些质量属性只是间接地促进用户需求的满足。

(2) 运行期质量属性。是软件系统在运行期间最终用户可以直接感受到的一类属性,这些质量属性直接影响用户对软件产品的满意度。

软件质量属性的内容如图 8.2 所示。

- 软件质量属性
  - 运行期质量属性
    - **性能**——软件系统及时提供相应服务的能力，表现在3个方面
      - 速度：以平均响应时间度量
      - 吞吐量：以单位时间处理的交易数度量
      - 持续高速度性：保持较高处理速度能力
    - **安全性**
      - 向合法用户提供服务
      - 阻止非法授权用户使用
      - 阻止恶意的攻击
    - **易用性**——软件系统易于使用的程度
    - **持续使用性**——系统长时间无故障运行的时间
    - **可伸缩性**——用户和数据量增加时，软件系统维持高服务质量的能力
    - **互操作性**——本软件系统和其他系统交换数据和相互调用服务的难易程度
    - **可靠性**——软件系统在一定的时间内无故障运行的能力
    - **鲁棒性**——也称健壮性、容错性，软件系统在以下情况下仍能正常运行的能力
      - 用户进行了非法操作
      - 相连的硬件系统发生故障
      - 发生其他非正常情况
  - 开发期质量属性
    - **易理解性**——设计被开发人员理解的难易程度
    - **可扩展性**——为适应新需求或需求的变化为软件增加功能的能力
    - **可重用性**——重用软件系统或其一部分的能力的难易程度
    - **可测试性**——对软件进行测试以证明其满足需求规约的难易
    - **可维护性**——为了达到以下目的之一而定位修改点并实施的难易程度：修改缺陷，增加功能，提高质量属性
    - **可移植性**——将软件系统从一个运行环境转移到另一个不同的运行环境的难易程度

图 8.2　软件质量属性

7. 测试文件的重要性主要体现在哪几个方面？

【答】 测试文件的重要性体现在如下几个方面：

（1）验证需求正确性。测试文件中规定了用以验证软件需求的测试条件，对弄清用户需求意图十分有益。

（2）检验测试资源。测试计划用文件的形式把测试过程确定下来，检验测试所需资源是否可用。

（3）明确任务风险。测试计划确定测试的内容，并给出测试任务的风险。

（4）生成测试用例。测试用例的好坏决定着测试工作的效率，测试用例的选择是确定若干测试数据及对应的预期测试结果，对于测试有非常关键的作用。

（5）评价测试结果。将测试结果与测试用例的预期结果进行比较，对已进行的测试提出评价意见。

(6) 再测试。测试文件规定的内容对维护阶段进行再测试非常有用。

(7) 决定测试有效性。测试文件能够为分析测试的有效性甚至整个软件的可用性提供依据。

8. 测试文档主要有哪些?

【答】 IEEE 给出的软件测试文档分为测试计划、测试设计规格说明、测试用例、规格说明、测试规程、测试日志、测试缺陷报告和测试总结报告。下面依次进行介绍。

(1) 软件测试计划文档。主要对软件测试项目以及所需要进行的测试工作、测试人员所应该负责的测试工作、测试过程、测试所需的时间和资源、测试风险、测试项通过/失败的标准、测试中断和恢复的规定、测试完成所提交的材料等做出预先的计划和安排。

(2) 软件测试设计规格说明文档。用于每个测试等级,以制定测试集的体系结构、通过/失败准则和覆盖跟踪。

(3) 软件测试用例规格说明文档。用于描述测试用例,包括测试项、输入规格说明、输出规格说明、预期要求和规程需求等。

(4) 测试规程。用于指定执行一个测试用例集的步骤。

(5) 测试日志。用于记录测试的执行情况,可根据需要选用。

(6) 软件缺陷报告。用来描述出现在测试过程或软件中的异常情况,这些异常情况可能存在于需求、设计、代码、文档或测试用例中。

(7) 测试总结报告。用于报告某个测试的完成情况,给出评价和建议。

9. 软件测试人员就是质量保证人员吗?

【答】 软件测试人员的职责是尽可能的找出软件缺陷,确保得以修复。而质量保证人员的主要职责是创建或者制定标准和方法,促进软件开发能力和减少软件缺陷。测试人员的主要工作是测试,质量保证人员日常工作的重要内容是检查与评审,测试工作也是质量保证人员的工作对象。软件测试和质量保证是相辅相成的关系,都是为了提高软件质量而工作。

10. 测试也有版本控制吗?

【答】 这里的版本控制主要是指测试对象的版本控制,也就是指对开发部提交的产品进行版本控制。在开发小组版本管理不规范的情况下,测试小组进行版本控制十分重要,要保证测试对象是可以控制的。建议开发和测试双方进行明确的约定,可以各自指定专门的测试版本负责人,制定提交原则,对提交情况进行详细的记录,这样就能基本避免版本失控导致的测试失误或无效。

11. 测试团队由哪些角色构成?这些角色的作用分别是什么?

【答】 软件测试团队中涉及的人员有测试经理、测试项目经理、测试设计员、测试员、测试系统管理员等,如表 8.1 所示。

表 8.1　测试团队构成及其职责

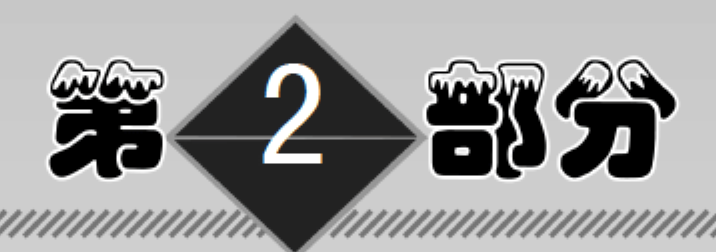
角　　色	具 体 职 责
测试经理	管理监督
测试项目经理	(1) 提供技术指导。 (2) 获取适当的资源。 (3) 提供管理报告
测试设计员	确定测试用例及其优先级并实施测试用例。 (1) 生成测试计划。 (2) 生成测试模型。 (3) 评估测试工作的有效性
测试员	(1) 执行测试。 (2) 记录测试结果。 (3) 记录变更请求
测试系统管理员	确保测试环境和资产得到管理和维护。 (1) 管理测试系统。 (2) 授予和管理角色对测试系统的访问权
数据库管理员	确保测试数据(数据库)环境得到管理和维护

12. 软件测试配置管理包括几个活动?

【答】 配置管理是建立和维护在软件生命周期中软件产品的完整性和一致性。一般来说,软件测试配置管理包括 4 个基本活动:配置标识、变更控制、配置状态报告、配置审计。

13. 软件风险管理一般分为几个步骤?

【答】 软件风险管理一般分成 5 个步骤,即风险识别、风险分析、风险计划、风险控制以及风险跟踪。



实 验 指 导

实验1

黑 盒 测 试

实验目的：

（1）理解黑盒测试的含义以及基本的测试方法。

（2）采用等价类划分法设计测试用例。

（3）采用边界值分析法设计测试用例。

（4）采用因果图设计测试用例。

实验环境：C语言、Java语言、Python语言等。

1.1 等价类划分法

1. 实验目的

掌握用等价类划分法设计测试用例的方法。

2. 实验内容

三角形问题：输入3个整数a、b、c，分别作为三角形的3条边，通过程序判断由3条边构成的三角形的类型为等边三角形、等腰三角形、一般三角形或者3条边不构成三角形。

3. 方案设计

输入3条边a、b、c必须满足以下条件：

条件1　$1 \leqslant a \leqslant 100$　　条件4　$a < b + c$

条件2　$1 \leqslant b \leqslant 100$　　条件5　$b < a + c$

条件3　$1 \leqslant c \leqslant 100$　　条件6　$c < a + b$

如果输入值a、b、c满足条件1、条件2和条件3，则输出下列4种情况之一：

（1）如果不同时满足条件4、条件5和条件6，则程序输出为“不构成三角形”。

（2）如果3条边相等，则程序输出为“等边三角形”。

（3）如果恰好有两条边相等，则程序输出为“等腰三角形”。

（4）如果3条边都不相等，则程序输出为“一般三角形”。

4. 测试数据及运行结果

等价类划分如表1.1所示。

表 1.1　等价类划分

输入条件	有效等价类	无效等价类
是否三角形的 3 条边	$(0<a<101)$ (1) $(0<b<101)$ (2) $(0<c<101)$ (3) $(a+b>c)$ (4) $(b+c>a)$ (5) $(c+a>b)$ (6)	$(a<=0 \| \| a>100)$ (7) $(b<=0 \| \| b>100)$ (8) $(c<=0 \| \| c>100)$ (9) $(a+b<=c)$ (10) $(b+c<=a)$ (11) $(c+a<=b)$ (12)
是否等腰三角形	$(a=b)$ (13) $(b=c)$ (14) $(a=c)$ (15)	$(a\neq b \ \&\& \ b\neq c \ \&\& \ a\neq c)$ (16)
是否等边三角形	$(a=b \ \&\& \ b=c \ \&\& \ a=c)$ (17)	$(a\neq b)$ (18) $(b\neq c)$ (19) $(a\neq c)$ (20)

根据等价类划分设计测试用例，如表 1.2 所示。

表 1.2　等价类测试用例

用例	(a,b,c)	覆盖等价类	输　出
Test1	(3,4,5)	(1)～(6)	一般三角形
Test2	(0,1,2)	(7)	不构成三角形
Test3	(2,0,1)	(8)	
Test4	(1,2,0)	(9)	
Test5	(1,2,3)	(10)	
Test6	(3,1,2)	(11)	
Test7	(1,3,2)	(12)	
Test8	(3,3,4)	(1)～(6),(13)	等腰三角形
Test9	(4,3,3)	(1)～(6),(14)	等腰三角形
Test10	(3,4,3)	(1)～(6),(15)	等腰三角形
Test11	(3,4,5)	(1)～(6),(16)	一般三角形
Test12	(3,3,3)	(1)～(6),(17)	等边三角形
Test13	(3,4,3)	(1)～(6),(18)	一般三角形
Test14	(3,4,3)	(1)～(6),(19)	一般三角形
Test15	(3,4,3)	(1)～(6),(20)	一般三角形

5. 源代码

```
package nan;
```

```
import java.util.Scanner;
public class Triangle_class {
    public static void main(String[] args){
        Scanner ss=new Scanner(System.in);
        float a=1,b=1,c=1;
        while(a!=1000||b!=1000||c!=1000){
        System.out.println("请输入三角形的三边 a,b,c 的值:");
        a=ss.nextFloat();
        b=ss.nextFloat();
        c=ss.nextFloat();
        System.out.println(test(a, b, c));
    }
}
public static String test(float x, float y, float z){
    String k=null;
    if(x>0 && x<101 && y>0 && y<101 && z>0 && z<101){
        if(x+y>z && x+z>y && y+z>x){
            if(x==y && y==z){
                k="是等边三角形";
            }
            if(x==y || x==z || y==z){
                k="是等腰三角形";
            }
            }
            else{
                k="是一般三角形";
            }
        }
        else {
            k="不构成三角形";
        }
        return k;
    }
}
```

1.2 边界值分析法

1. 实验目的

掌握用边界值分析法设计测试用例的方法。

2. 实验内容

现有一个学生标准化考试批阅试卷、产生成绩报告的程序。其规格说明为：程序的

输入文件由80个字符的记录组成，分为试题部分和学生答卷部分，具体内容如图1.1所示。

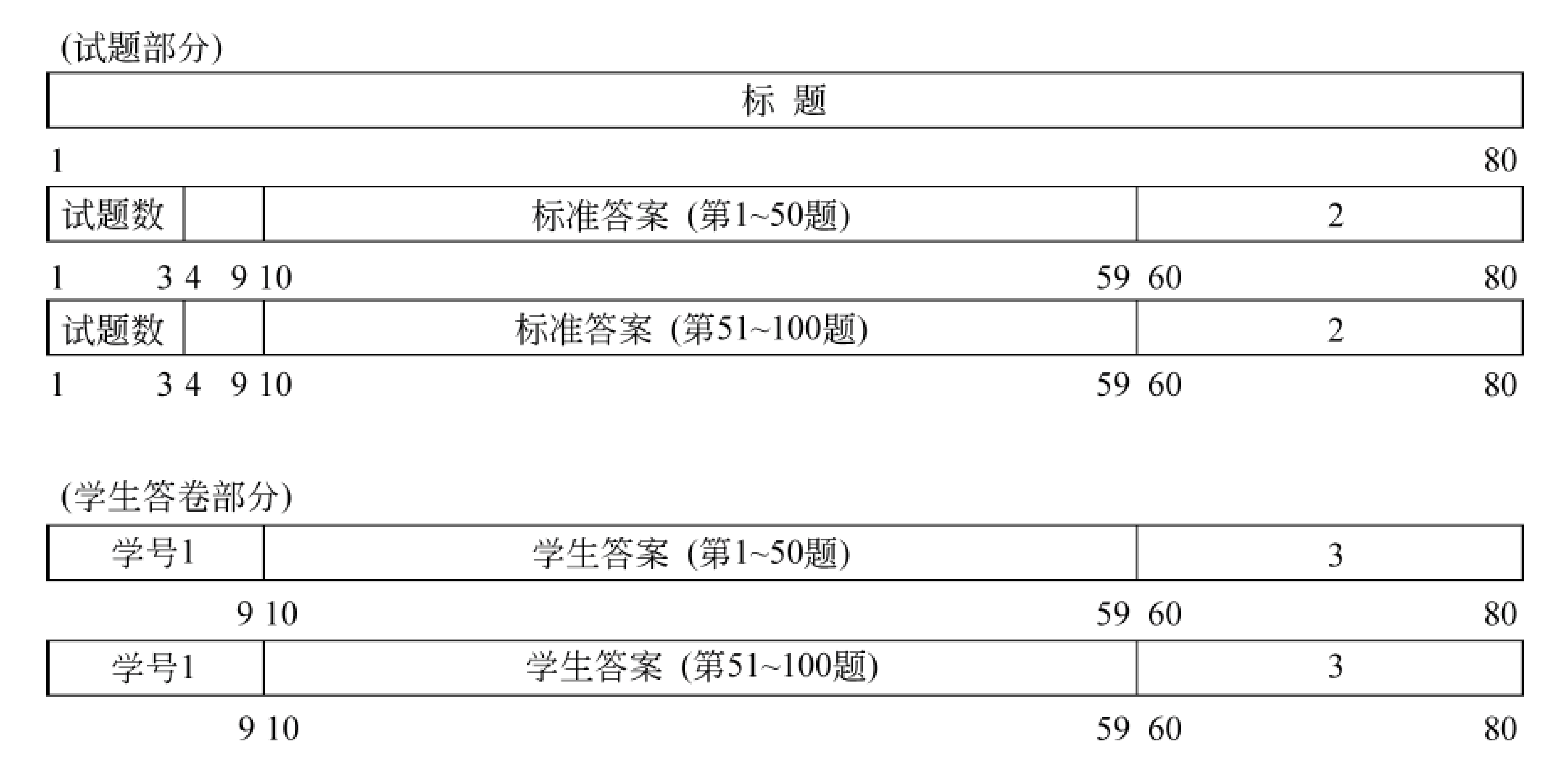


图1.1 输入文件组成

试题部分包括如下内容：

(1) 标题。这一组只有一个记录，其内容为输出成绩报告的名字。

(2) 标准答案。每个记录在第80个字符处标以数字2。第一个记录的第1～3个字符为“试题数”，用于标示题目编号；第10～59个字符给出第1～50题的答案，以此类推。

学生答卷部分包括内容：每个记录的第80个字符均为数字3。第1～9个字符给出学号，第10～59字符给出第1至第50题的答案，以此类推。

程序的输出有4个报告：

(1) 按学号排列的成绩单，列出每个学生的成绩、名次。

(2) 按学生成绩排序的成绩单。

(3) 平均分数及标准偏差的报告。

(4) 试题分析报告。按试题号排序，列出各题学生答对的百分比。

3. 测试数据及运行结果

表1.3为输入条件及相应的测试用例。

表1.3 输入条件及相应的测试用例

输入条件	测试用例
输入文件	空输入文件
标题	没有标题 标题只有一个字符 标题有80个字符

续表

输入条件	测试用例
试题数	试题数为 1 试题数为 50 试题数为 51 试题数为 100 试题数为 0 试题数含有非数字字符
标准答案记录	没有标准答案记录,有标题 标准答案记录多于一个 标准答案记录少于一个
学生人数	0 个学生 1 个学生 200 个学生 201 个学生
学生答题	某学生只有一个回答记录,但有两个标准答案记录 该学生是文件中的第一个学生 该学生是文件中最后一个学生(记录数出错的学生)
学生答题	某学生有两个回答记录,但只有一个标准答案记录 该学生是文件中的第一个学生(记录数出错的学生) 该学生是文件中最后一个学生
学生成绩	所有学生的成绩都相同 所有学生的成绩各不相同 部分学生的成绩相同 (检查是否能按成绩正确排名次) 有一个学生得 0 分 有一个学生得 100 分

输出条件及相应的测试用例如表 1.4 所示。

表 1.4 输出条件及相应的测试用例

输出条件	测试用例
输出第(1)、(2)个报告	有一个学生的学号最小(检查按学号排序是否正确) 有一个学生的学号最大(检查按学号排序是否正确) 适当的学生人数,使产生的报告刚好打满一页(检查打印页数) 学生人数比上一用例多出 1 人(检查打印换页)
输出第(3)个报告	平均成绩 100 平均成绩 0 标准偏差为最大值(有一半为 0 分,其他为 100 分) 标准偏差为 0(所有成绩相同)
输出第(4)个报告	所有学生都答对了第一题 所有学生都答错了第一题 所有学生都答对了最后一题 所有学生都答错了最后一题 选择适当的试题数,使第(4)个报告刚好打满一页 试题数比上一用例多 1,使报告打满一页后,刚好剩下一题未打

4. 源代码

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#define LineMax 80
#define FilePathMax 80
#define perPageNumber 50
#define startAnswer 10
#define ExamMaxNumber 999
typedef struct FileNode
{
    char line[LineMax+1];
    struct FileNode * next;
}FileNode;
typedef struct Student{
    struct Student *  next;
    char id[4];
    char name[7];
    int score;
    int position;
}Student;
typedef struct ExamPaper{
    char examPaperName[LineMax+1];
    int examPagerRightNumber[ExamMaxNumber];
    int examNumber;
}ExamPaper;
int studentNumber=0;
FileNode *  ReadFile()
{
    FileNode * head, * p, * q;
    char FilePath[20];
    FILE * fp;
    head= (FileNode * )malloc(sizeof(FileNode));
    head ->next=NULL;
    p=head;
    printf("请输入正确的文件地址\n");
    gets(FilePath);
    fp=fopen(FilePath,"r");
    if(fp==NULL){
```

```
        printf("文件路径输入错误!!!\n");
        exit(-1);
    }else{
        printf("文件路径输入正确,正在解析\n");
    }
    while(!feof(fp))
    {
        q=(FileNode *)malloc(sizeof(FileNode));
        if(fgets(q->line,LineMax+1,fp)==NULL)
        {
            free(q);
            break;
        }
        p->next=q;
        p=q;
    }
    fclose(fp);
    p->next=NULL;
    p=head->next;
    free(head);
    return p;
}
FileNode * GetStudentNode(FileNode * node)
{
    FileNode * p=node, * q=node->next;
    while(q)
    {
        if(q->line[LineMax-1]=='3')
        {
            p->next=NULL;
            return q;
        }
        p=q;
        q=q->next;
    }
    return q;
}
int stringToInteger(char * string,int start,int end){
    int i=start,tem=0;
    for(; i<=end ; i++){
        tem=tem * 10;
        tem=tem+string[i]-'0';
    }
```

```
    return tem;
}
int getPageNumber(char * array)
{
    int examnum=stringToInteger(array,0,2);
    return(examnum/perPageNumber)+(((examnum%perPageNumber)>0)   1 : 0);
}
void subStringForOther(char * substring,int start,int end,char * other){
    int i;
    for(i=start ; i <=end ; i++){
        other[i-start]=substring[i];
    }
    other[i-start]='\0';
}
Student * produceStudentNews(FileNode * student,FileNode * answer,ExamPaper * examPage)
{
    FileNode * p=answer, * q=student, * pq;
    Student * students=(Student * )malloc(sizeof(Student)), * pstudents, * qstudents;
    int page,i,j;
    if(p ==NULL)
    {
        printf("没有标题,没有试卷\n");
        return NULL;
    }
    else
    {
        if(p ->line[LineMax-1] =='2')
        {
            printf("没有标题\n");
            return NULL;
        }
        else if(p->next==NULL)
        {
            printf("没有试卷\n");
            return NULL;
        }
        strcpy(examPage->examPaperName, p ->line);
        p=p ->next;
    }
    students ->next=NULL;
    pstudents=students;
    page=getPageNumber(p->line);
    examPage ->examNumber=stringToInteger(p->line,0,2);
```

```
    for(i=0 ; i<ExamMaxNumber ; i++)
    {
        examPage->examPagerRightNumber[i]=0;
    }
    while(q)
    {
        p=answer ->next;
        qstudents=(Student *)malloc(sizeof(Student));
        qstudents ->score=0;
        subStringForOther(q ->line,0,2,qstudents->id);
        subStringForOther(q ->line,3,8,qstudents->name);
        for(i=0; i<page ; i++)
        {
            for(j=0 ; j<perPageNumber ; j++)
            {
                if(p ->line[9+j]  ==q ->line[9+j])
                {
                    qstudents ->score++;
                    examPage ->examPagerRightNumber[j+perPageNumber * i-1]++;
                }
            }
            p=p ->next;
            q=q ->next;
        }
        pstudents ->next=qstudents;
        pstudents=qstudents;
        studentNumber++;
    }
    pstudents ->next=NULL;
    pstudents=students->next;
    free(students);
    if(pstudents==NULL)
    {
        printf("没有学生\n");
    }
    return pstudents;
}
void printStudentNews(Student * students){
    while(students){
        printf("学号:%s 姓名:%s 成绩:%d 名次:%d\n",students->id,students->name,
        students->score,students->position);
        students=students->next;
    }
```

```
}
Student * setPosition(Student * students){
    Student * p;
    int i=0,j;
    Student **ss,tem;
    if(students==NULL){
        return  NULL;
    }
    ss=(Student **)malloc(studentNumber * sizeof(Student * ));
    for(p=students; p ; p=p ->next,i++){
        ss[i]=p;
    }
    for(i=0 ; i<studentNumber ; i++){
        for(j=i+1 ; j<studentNumber ; j++){
            if(ss[i]->score<ss[j]->score){
                tem= * ss[i];
                * ss[i]= * ss[j];
                * ss[j]=tem;
            }
        }
        (* ss[i]).position=i+1;
    }
    for(i=0 ; i<studentNumber-1 ; i++){
        ss[i]->next=ss[i+1];
    }
    ss[i]->next=NULL;
    students=ss[0];
    free(ss);
    return students;
}
Student * sortByScore(Student * students){
    Student * p;
    int i=0,j;
    Student * * ss,tem;
    printf("按分数排序:\n");
    if(students==NULL){
        printf("无学生信息\n");
        return  NULL;
    }
    ss=(Student * * )malloc(studentNumber * sizeof(Student * ));
    for(p=students; p ; p=p ->next,i++){
        ss[i]=p;
    }
```

```
    for(i=0 ; i<studentNumber ; i++){
        for(j=i+1 ; j<studentNumber ; j++){
            if(ss[i]->score<ss[j]->score){
                tem=*ss[i];
                *ss[i]=*ss[j];
                *ss[j]=tem;
            }
        }
    }
    for(i=0 ; i<studentNumber-1 ; i++){
        ss[i]->next=ss[i+1];
    }
    ss[i]->next=NULL;
    students=ss[0];
    free(ss);
    printStudentNews(students);
    return students;
}
Student * sortById(Student * students){
    Student * p;
    int i=0,j;
    Student * * ss,tem;
    printf("按学号排序:\n");
    if(students==NULL){
        printf("无学生信息\n");
        return NULL;
    }
    ss=(Student * * )malloc(studentNumber * sizeof(Student * ));
    for(p=students; p ; p=p ->next,i++){
        ss[i]=p;
    }
    for(i=0 ; i<studentNumber ; i++){
        for(j=i+1 ; j<studentNumber ; j++){
            if(strcmp(ss[i]->id,ss[j]->id)>0){
                tem=*ss[i];
                *ss[i]=*ss[j];
                *ss[j]=tem;
            }
        }
    }
    for(i=0 ; i<studentNumber-1 ; i++){
        ss[i]->next=ss[i+1];
    }
```

```
        ss[i]->next=NULL;
        students=ss[0];
        printStudentNews(students);
        free(ss);
        return students;
    }
    double getAverage(Student * students){
        int i;
        double scoreAll=0;
        if(students==NULL)return 0.0;
        for(i=0 ; i<studentNumber ; i++){
            scoreAll +=students ->score;
            students=students ->next;
        }
        return scoreAll/studentNumber;
    }
    double getStandardDeviation(Student * students,double average){
        double tem=0,tem_single=0;
        int i=0 ;
        if(students==NULL)return 0.0;
        for(;i<studentNumber ; i++){
            tem_single=students ->score-average;
            tem +=tem_single * tem_single;
            students=students ->next;
        }
        return sqrt(tem/studentNumber);
    }
    void printExamNews(Student * students,FileNode * answer){
        double average=getAverage(students),standardDeviation;
        standardDeviation=getStandardDeviation(students,average);
        printf("试卷名称:%s \n 平均成绩:%lf 标准偏差:%lf\n",answer -> line ,average,
        standardDeviation);
    }
    void ExamAnalysisReport(ExamPaper examPage){
        int i=0;
        examPage.examPaperName[LineMax]='\0';
        printf("试卷名称:%s\n",examPage.examPaperName);
        for(; i<examPage.examNumber; i++){
            printf("第%d题正确率:%lf\n",i+1,examPage.examPagerRightNumber[i]/(examPage.
            examNumber * 1.0));
        }
    }
    void FreeStudentNode(FileNode * node){
```

```
    FileNode * p=node;
    while(node){
        p=node->next;
        free(node);
        node=p;
    }
}
void printStudent(FileNode * student){
    while(student){
        printf("%s\n",student->line);
        student=student->next;
    }
}
int main(void)
{
    FileNode * student=NULL, * answer=NULL;
    Student * students;
    ExamPaper examPage;
    answer=ReadFile();
    student=GetStudentNode(answer);
    students=produceStudentNews(student,answer,&examPage);
    if(students!=NULL)
    {
        students=setPosition(students);
        students=sortByScore(students);
        students=sortById(students);
        printExamNews(students,answer);
        ExamAnalysisReport(examPage);
    }
    if(student!=NULL)
    {
        FreeStudentNode(student);
    }
    if(answer!=NULL)
    {
        FreeStudentNode(answer);
    }
}
```

程序运行结果如图 1.2 所示。

```
选择E:\学习\2222.exe
Please input the exam name:
啊
Please input the students number:
10
Please input the exam number:
16
文件路径输入正确. 正在解析
按分数排序:
学号:007 姓名:qglamm 成绩:41 名次:1
学号:005 姓名:nohbun 成绩:40 名次:2
学号:001 姓名:hcrsmh 成绩:39 名次:3
学号:003 姓名:bhnval 成绩:39 名次:4
学号:002 姓名:dbsryg 成绩:38 名次:5
学号:000 姓名:uheske 成绩:38 名次:6
学号:004 姓名:ktlujq 成绩:37 名次:7
学号:006 姓名:alqtay 成绩:36 名次:8
学号:009 姓名:esubtj 成绩:36 名次:9
学号:008 姓名:zmjpuu 成绩:35 名次:10
按学号排序:
学号:000 姓名:uheske 成绩:38 名次:6
学号:001 姓名:hcrsmh 成绩:39 名次:3
学号:002 姓名:dbsryg 成绩:38 名次:5
学号:003 姓名:bhnval 成绩:39 名次:4
学号:004 姓名:ktlujq 成绩:37 名次:7
学号:005 姓名:nohbun 成绩:40 名次:2
学号:006 姓名:alqtay 成绩:36 名次:8
学号:007 姓名:qglamm 成绩:41 名次:1
学号:008 姓名:zmjpuu 成绩:35 名次:10
学号:009 姓名:esubtj 成绩:36 名次:9
试卷名称:啊
平均成绩:37.900000 标准偏差:1.813836
试卷名称:啊
第1题正确率:0.062500
搜狗拼音输入法 全 :0
第3题正确率:0.187500
第4题正确率:0.125000
第5题正确率:0.250000
第6题正确率:0.062500
第7题正确率:0.062500
第8题正确率:0.250000
第9题正确率:0.062500
第10题正确率:0.062500
第11题正确率:0.187500
第12题正确率:0.250000
第13题正确率:0.187500
第14题正确率:0.187500
第15题正确率:0.187500
第16题正确率:0.625000
```

图 1.2　程序运行结果

1.3　因　果　图

1. 实验目的

掌握用因果图设计测试用例的方法。

2. 实验内容

售货机软件若投入 1.5 元硬币，按“可乐”“雪碧”或“红茶”按钮，送出相应的饮料；若投入的是 2 元硬币，在送出饮料的同时退还 5 角硬币。请用因果图设计测试用例。

3. 测试数据及运行结果

步骤 1：原因和结果分析。

原因(输入)：

- (1)投入 1.5 元硬币。
- (2)投入 2 元硬币。
- (3)按“可乐”按钮。
- (4)按“雪碧”按钮。

- (5)按“红茶”按钮。

中间状态：

- (11)已投币。
- (12)已按钮。

结果(输出)：

- (21)退还 5 角硬币。
- (22)送出可乐。
- (23)送出雪碧。
- (24)送出红茶。

步骤 2：画出因果图，如图 1.3 所示。

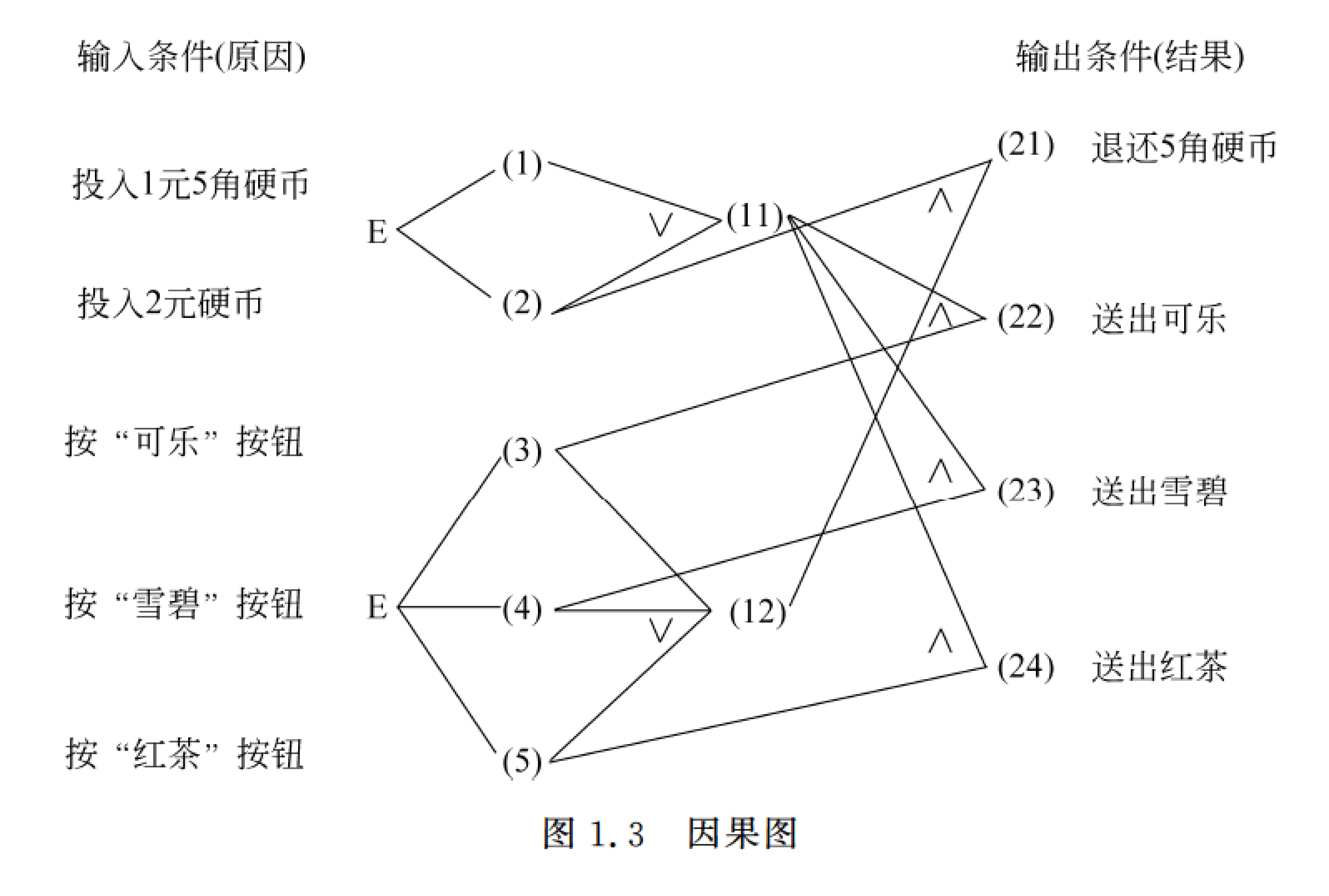


图 1.3　因果图

步骤 3：设计决策表。

根据因果图设计决策表，如表 1.5 所示，这里只有 11 个，而不是 $2^5=32$ 个，这是因为图 1.3 中有很多限制条件导致某些情况不可能出现。

表 1.5　决策表

条件桩			动作桩										
			1	2	3	4	5	6	7	8	9	10	11
输入	投入 1.5 元硬币	(1)	1	1	1	1	0	0	0	0	0	0	0
	投入 2 元硬币	(2)	0	0	0	0	1	1	1	1	0	0	0
	按“可乐”按钮	(3)	1	0	0	0	1	0	0	0	1	0	0
	按“雪碧”按钮	(4)	0	1	0	0	0	1	0	0	0	1	0
	按“红茶”按钮	(5)	0	0	1	0	0	0	1	0	0	0	1
中间节点	已投币	(11)	1	1	1	1	1	1	1	1	0	0	0
	已按钮	(12)	1	1	1	0	1	1	1	0	1	1	1

续表

条件桩			动作桩										
			1	2	3	4	5	6	7	8	9	10	11
输出	退还5角硬币	(21)	0	0	0	0	1	1	1	0	0	0	0
	送出可乐	(22)	1	0	0	0	1	0	0	0	0	0	0
	送出雪碧	(23)	0	1	0	0	0	1	0	0	0	0	0
	送出红茶	(24)	0	0	1	0	0	0	1	0	0	0	0

步骤4：设计测试用例。

根据决策表设计测试用例，如表1.6所示。

表1.6 测试用例

用例编号	用例说明	输入数据	预期结果
01	投入硬币，按下按钮	1.5元，按“可乐”按钮	送出可乐
02	投入硬币，按下按钮	1.5元，按“雪碧”按钮	送出雪碧
03	投入硬币，按下按钮	1.5元，按“红茶”按钮	送出红茶
04	投入硬币	1.5元	给出提示信息
05	投入硬币，按下按钮	2元，按“可乐”按钮	退还5角，送出可乐
06	投入硬币，按下按钮	2元，按“雪碧”按钮	退还5角，送出雪碧
07	投入硬币，按下按钮	2元，按“红茶”按钮	退还5角，送出红茶
08	投入硬币	2元	给出提示信息
09	按下按钮	按“可乐”按钮	给出提示信息
10	按下按钮	按“雪碧”按钮	给出提示信息
11	按下按钮	按“红茶”按钮	给出提示信息

4. 源代码

```
#include<stdio.h>
#include<string.h>
double goodsCost[]={1.5,1.5,1.5};
char * goodsName[]={"可乐", "雪碧", "红茶"};
int IsGoodsExist(int type)
{
    if(type>=0&&type<3)
    return 1;
    return 0;
}
void PrintExistGoodsType()
```

```
{
    int i=0;
    for(i=0; i<3;i++)
    {
        printf("编号:%d 名称:%s 价格:%lf\n",i,goodsName[i],goodsCost[i]);
    }
}
double scanInputMoney()
{
    double inputmoney;
    int isMoneySupport;
    do{
        printf("请输入投入硬币数目:\n");
        scanf("%lf",&inputmoney);
        if(inputmoney==1.5||inputmoney==2.0)
            {
                isMoneySupport=1;
            }
            else
            {
                isMoneySupport=0;
            }
        }
        while(!isMoneySupport);
        return inputmoney;
}
int scanInputGoodsType()
{
    int type;
    do{
        printf("请选择你要的类型:\n");
        printf("请输入编号\n");
        PrintExistGoodsType();
        scanf("%d",&type);
        if(IsGoodsExist(type)==0){
            printf("你要购买的类型不存在,请重新选择\n");
        }
    }while(!IsGoodsExist(type));
    return type;
}
int TryBuyIt(double * inputmoney,int type)
{
    if(* inputmoney>=goodsCost[type])
    {
```

```
            * inputmoney -=goodsCost[type];
            return 1;
        }
        else
        {
        printf("投入硬币不足,请重新投入\n");
        }
        return 0;
}
void BuyIt(double * inputmoney,int type){
int tryit;
tryit=TryBuyIt(inputmoney,type);
    while(tryit==0){
        * inputmoney=scanInputMoney();
        tryit=TryBuyIt(inputmoney,type);
    }
}
void ReturnCoin(double inputmoney){
    if(inputmoney!=0.0){
        printf("返回的剩余零钱为:%lf\n",inputmoney);
    }
}
void ReturnGoods(int type){
    printf("你购买的货物为%s\n",goodsName[type]);
}
void Handle(double inputmoney,int type){
    ReturnCoin(inputmoney);
    ReturnGoods(type);
}
int main(void)
{
    double inputmoney;
    int type;
    inputmoney=scanInputMoney();
    type=scanInputGoodsType();
    BuyIt(&inputmoney,type);
    Handle(inputmoney,type);
    return 0;
}
```



白 盒 测 试

实验目的：

（1）理解白盒测试的含义以及基本的测试方法。

（2）掌握逻辑覆盖的语句覆盖、判定覆盖、条件覆盖等方法。

（3）掌握路径分析方法。

实验环境：C 语言、Java 语言、Python 语言等。

2.1 逻 辑 覆 盖

1. 实验目的

（1）理解逻辑覆盖测试方法。

（2）采用语句覆盖、判定覆盖、条件覆盖设计测试用例。

2. 实验内容

被测试的程序段如下：

```
begin
s1;
if  (x=0)  and  (y>2)
then  s2;
if  (x<1)  or  (y=1)
then  s3;
s4;
end
```

采用语句覆盖、条件覆盖和判定覆盖设计测试用例。

3. 测试数据及运行结果

程序段的流程图如图 2.1 所示。

（1）语句覆盖。设计若干测试用例，运行被测程序，使程序中每个可执行语句至少执行一次。

x=0，y=3 满足语句覆盖。

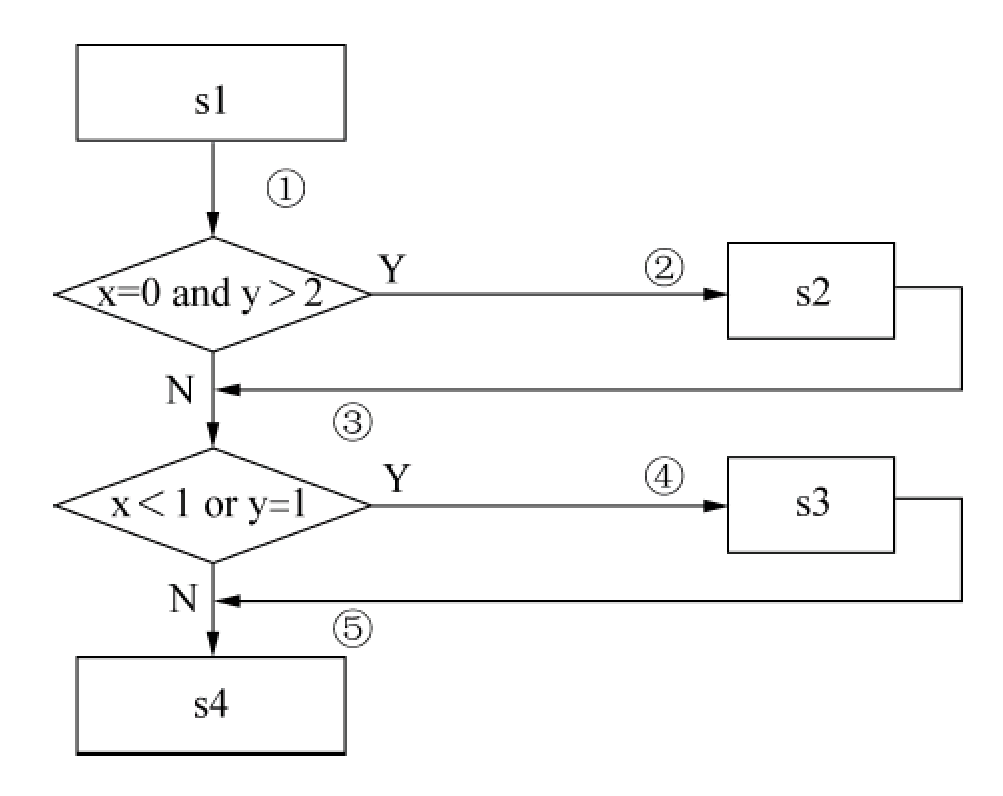


图 2.1 程序流程图

(2) 判定覆盖。设计若干测试用例,运行被测程序,使得程序中每个判断的取真分支和取假分支至少经历一次,即判断真假值均能被满足。

x=0,y=3 和 x=1,y=2 满足判定覆盖。

(3) 条件覆盖。设计若干测试用例,运行被测程序,使每个判断中每个条件的可能取值至少满足一次。

x=0,y=3 和 x=3,y=1 满足条件覆盖。

2.2 路径分析

1. 实验目的

(1) 理解路径分析方法。

(2) 采用路径分析方法设计测试用例。

2. 实验内容

程序流程图如图 2.2 所示,本程序最多输入 50 个值(以 −1 作为输入结束标志),计算其中有效的学生分数的个数、总分数和平均值。

3. 测试数据及运行结果

步骤 1:导出程序流程图的控制流图,如图 2.3 所示。

步骤 2:确定环形复杂性度量 $V(G)$。

(1) $V(G)=6$(个区域)。

(2) $V(G)=E-N+2=16-12+2=6$,其中 E 为流图中的边数,N 为结点数。

(3) $V(G)=P+1=5+1=6$,其中 P 为谓词结点的个数。在控制流图中,结点 2、3、5、6、9 是谓词结点。

步骤 3:确定基本路径集合(即独立路径集合)。于是可确定 6 条独立的路径。

路径 1:1-2-9-10-12。

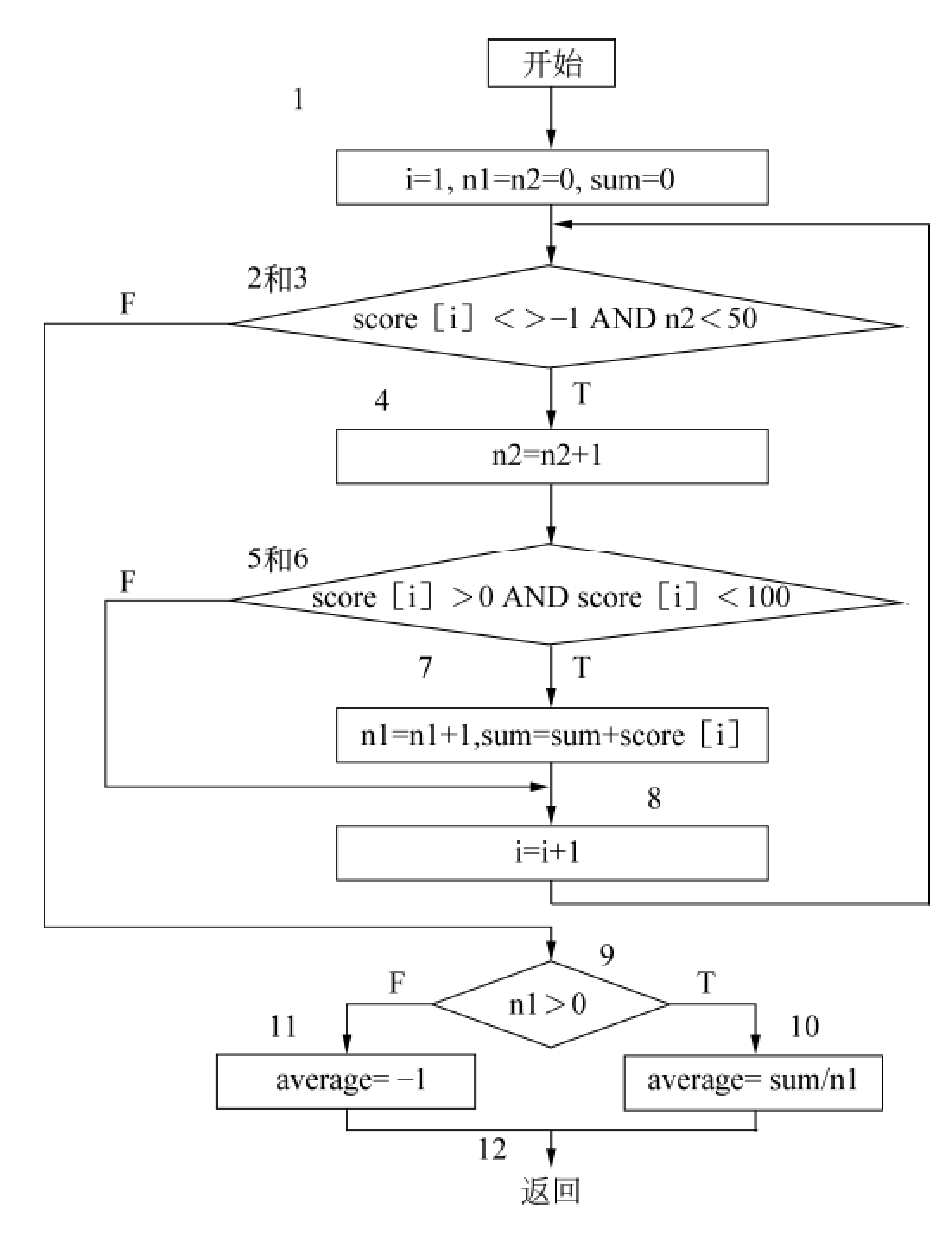


图 2.2　程序流程图

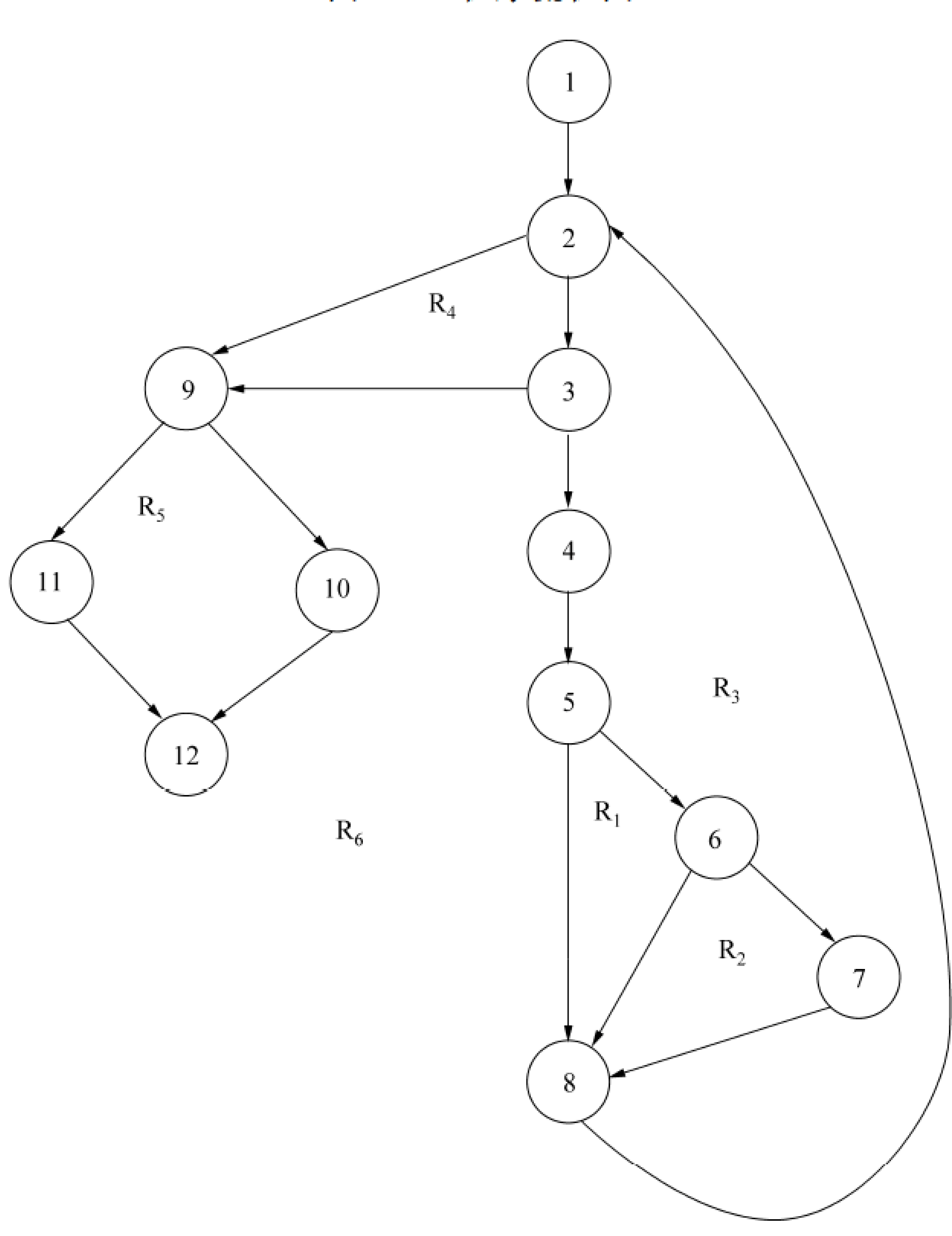


图 2.3　控制流图

路径 2：1-2-9-11-12。

路径 3：1-2-3-9-10-12。

路径 4：1-2-3-4-5-8-2…。

路径 5：1-2-3-4-5-6-8-2…。

路径 6：1-2-3-4-5-6-7-8-2…。

步骤 4：为每一条独立路径各设计一组测试用例，以便强迫程序沿着该路径至少执行一次。

(1) 路径 1(1-2-9-10-12)的测试用例：

score[k]=有效分数值($k<i$)。

score[i]=−1($2\leqslant i\leqslant 50$)。

期望结果：根据输入的有效分数算出正确的分数个数 n1、总分 sum 和平均分 average。

(2) 路径 2(1-2-9-11-12)的测试用例：

score[1]=−1。

期望的结果：average=−1，其他量保持初始值。

(3) 路径 3(1-2-3-9-10-12)的测试用例：

输入多于 50 个有效分数，即试图处理 50 个以上的分数，要求前 51 个为有效分数。

期望结果：n1=50 且算出正确的总分和平均分。

(4) 路径 4(1-2-3-4-5-8-2…)的测试用例：

score[i]=有效分数($i<50$)。

score[k]<0，$k<i$。

期望结果：根据输入的有效分数算出正确的分数个数 n1、总分 sum 和平均分 average。

4. 源代码

```
main()
{
  int i=1,n1=n2=0;
  float sum=0;
  float average;
  float score[100];
  while(score[i]!=-1 && n2<50)
  {
     n2=n2+1;
     if(score[i]>0&&score[i]<100)
     {
       n1=n1+1;
       sum=sum+score[i];
     }
     i=i+1;
```

```
    }
    if(n1>0)
        average=sum/n1;
    else
        average=-1;
    printf("n1=%d,sum=%f,average=%f\n",n1,sum,average);
}
```



单元测试软件 JUnit

实验目的：

（1）理解 JUnit 的断言等内容。

（2）测试 Calculator 类。

（3）测试 Sorting 类。

（4）测试 WordDealUtil 类。

（5）测试 Triangle 类。

实验环境：JUnit。

3.1 JUnit 介绍

3.1.1 JUnit 特点

JUnit 是由 Erich Gamma 和 Kent Beck 用 Java 编写的进行单元测试的开源框架，JUnit 测试是程序员测试，程序员需要知道被测软件如何（How）完成功能和完成什么样（What）的功能。开发者只需遵循 JUnit 的框架编写测试代码，JUnit 就可以自动完成测试。由于 JUnit 相对独立于所编写的代码，测试代码可以先于实现代码进行编写，符合极限编程的测试优先设计的理念。

JUnit 共有 7 个包，核心的包就是 JUnit.framework 和 JUnit.runner。其中，JUnit.framework 包负责整个测试对象的构架，JUnit.runner 负责测试驱动。

JUnit 有 TestCase、TestResult、TestSuite、TestRunner 4 个重要的类。

- TestCase 类负责测试时对类进行初始化以及测试方法调用。
  书写测试方法：public void testXXXXX()。
- TestResult 负责收集 TestCase 的执行结果。
- TestSuite 用于包装 TestCase。
- TestRunner 负责对整个测试流程的跟踪，用于显示测试结果，报告测试的进度。

JUnit 用于单元级测试，具有如下优势：

（1）JUnit 完全免费。JUnit 完全开放源代码，在其基础上可以进行二次开发。

（2）使用方便。JUnit 可以快速撰写测试并检测程序代码，JUnit 执行测试类似于编译程序。

（3）JUnit 检验结果并提供立即回馈。JUnit 自动执行并且检查结果，执行测试时立即回馈信息。

（4）JUnit 合成测试系列的层级架构。JUnit 引入了重构概念，把测试组织成测试组，允许组合多个测试自动回归测试。

（5）与 IDE 的集成。JUnit 一般集成在 Eclipse 软件工具中个，使用十分方便。

3.1.2 JUnit 断言

JUnit 通过断言来判断语句是否正确。可以使用的断言如下：

- assertTrue(String message, boolean condition)
- assertTrue(boolean condition)
- assertFalse(String message, boolean condition)
- assertFalse(boolean condition)
- assertEquals(String message, Object expected, Object actual)
- assertEquals(Object expected, Object actual)
- assertEquals(String message, String expected, String actual)
- assertEquals(String expected, String actual)
- assertEquals(String message, double expected, double actual, double delta)
- assertEquals(double expected, double actual, double delta)
- assertEquals(String message, float expected, float actual, float delta)
- assertEquals(floatexpected, float actual, float delta)
- assertNotNull(String message, Object object)
- assertNotNull(Object object)
- assertNull(String message, Object object)
- assertNull(Object object)

3.2 测试 Calculator 类

3.2.1 Calculator 类

Calculator 类实现了加、减、乘、除四则运算。

Calculator 类代码如下：

```
package andycpp;
public class Calculator {
    private static int result;              //静态变量用于存储运行结果
    public void add(int n){
        result=result+n;
    }
    public void substract(int n){
        result=result-1;                    //Bug: 正确的应该是 result =result-n
```

```
        }
        public void multiply(int n){
        }                                          //此方法尚未写好
        public void divide(int n){
            result=result /1;                      //Bug: 正确的应该是 result =result/n
        }
        public void clear(){                       //将结果清零
            result=0;
        }
        public int getResult(){
            return result;
        }
}
```

3.2.2 CalculatorTest 类

步骤 1：将 JUnit 4 单元测试包引入 JUnit_Test 项目。在该项目上右击，选择“属性”命令，如图 3.1 所示。

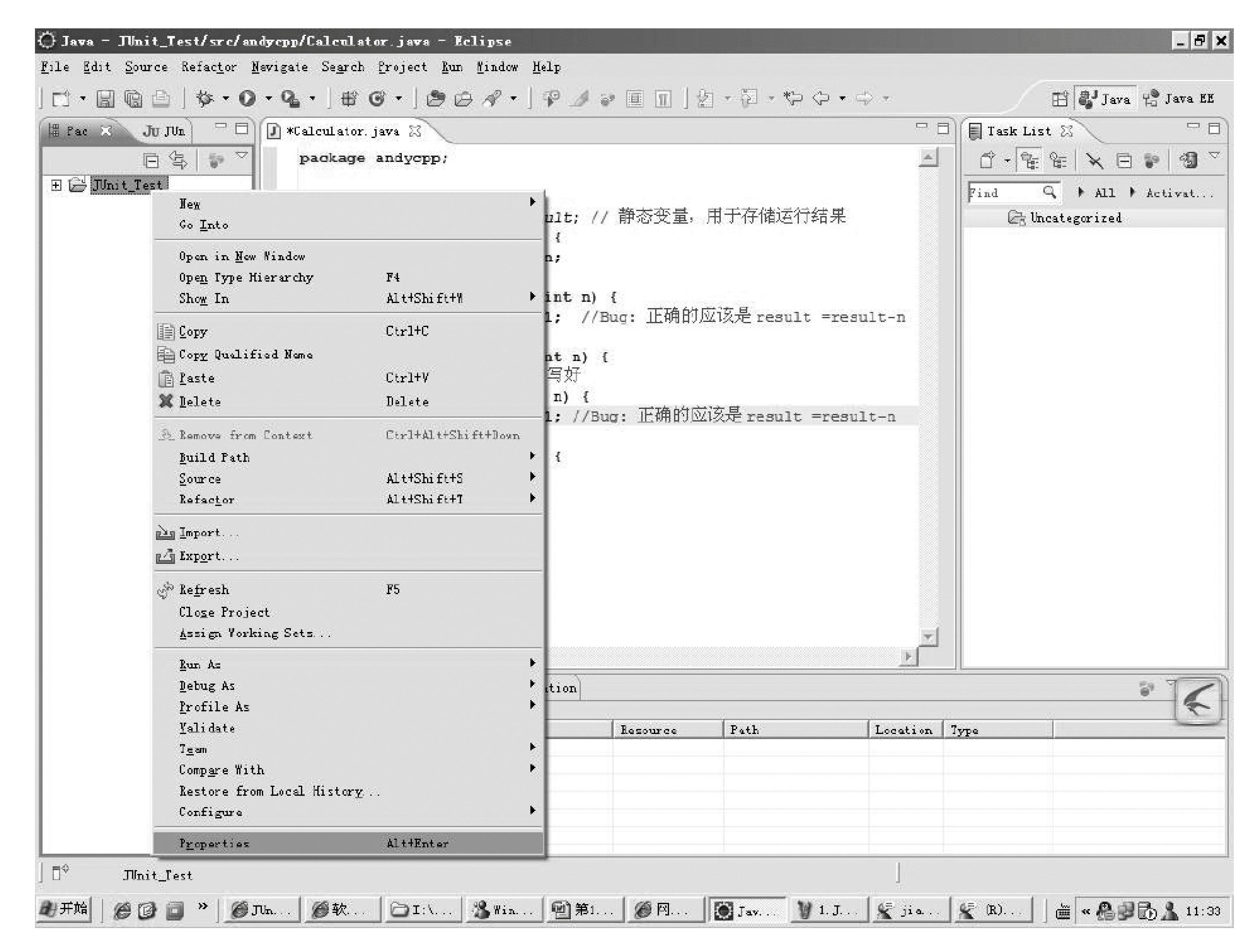


图 3.1 将 JUnit 4 测试包引入 JUnit_Test 项目截图 1

在弹出的属性对话框中，选择 Java Build Path，选择 Libraries 标签，单击 Add Library 按钮，如图 3.2 所示，选择 JUnit 4，将 JUnit 4 软件包加入到 JUnit_Test 项目。

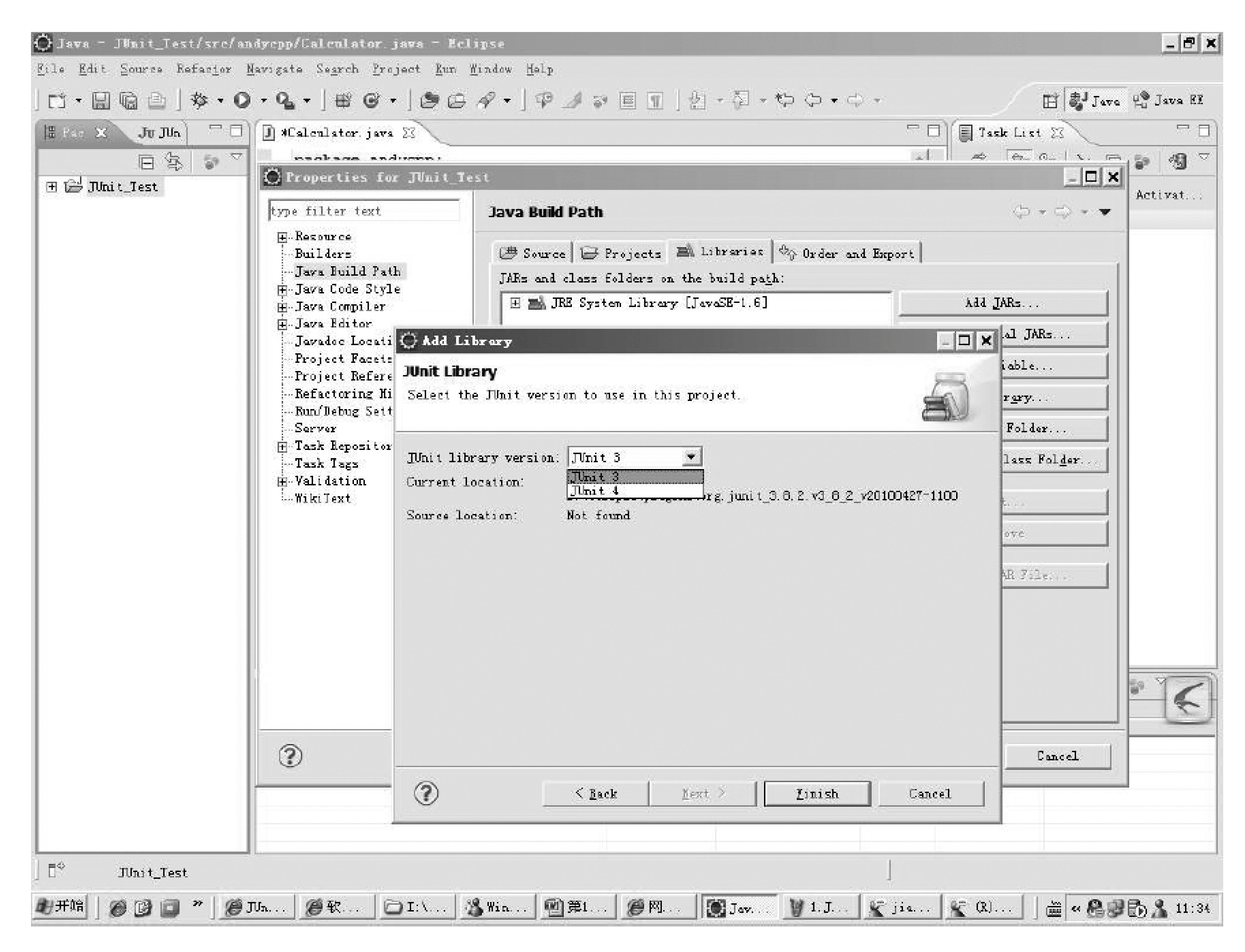


图 3.2　将 JUnit 4 测试包引入 JUnit_Test 项目截图 2

步骤 2：生成 JUnit 测试框架。在 Eclipse 的 Package Explorer 中右击 Calculator 类，在弹出菜单中选择 New→JUnit Test Case 命令，如图 3.3 所示。

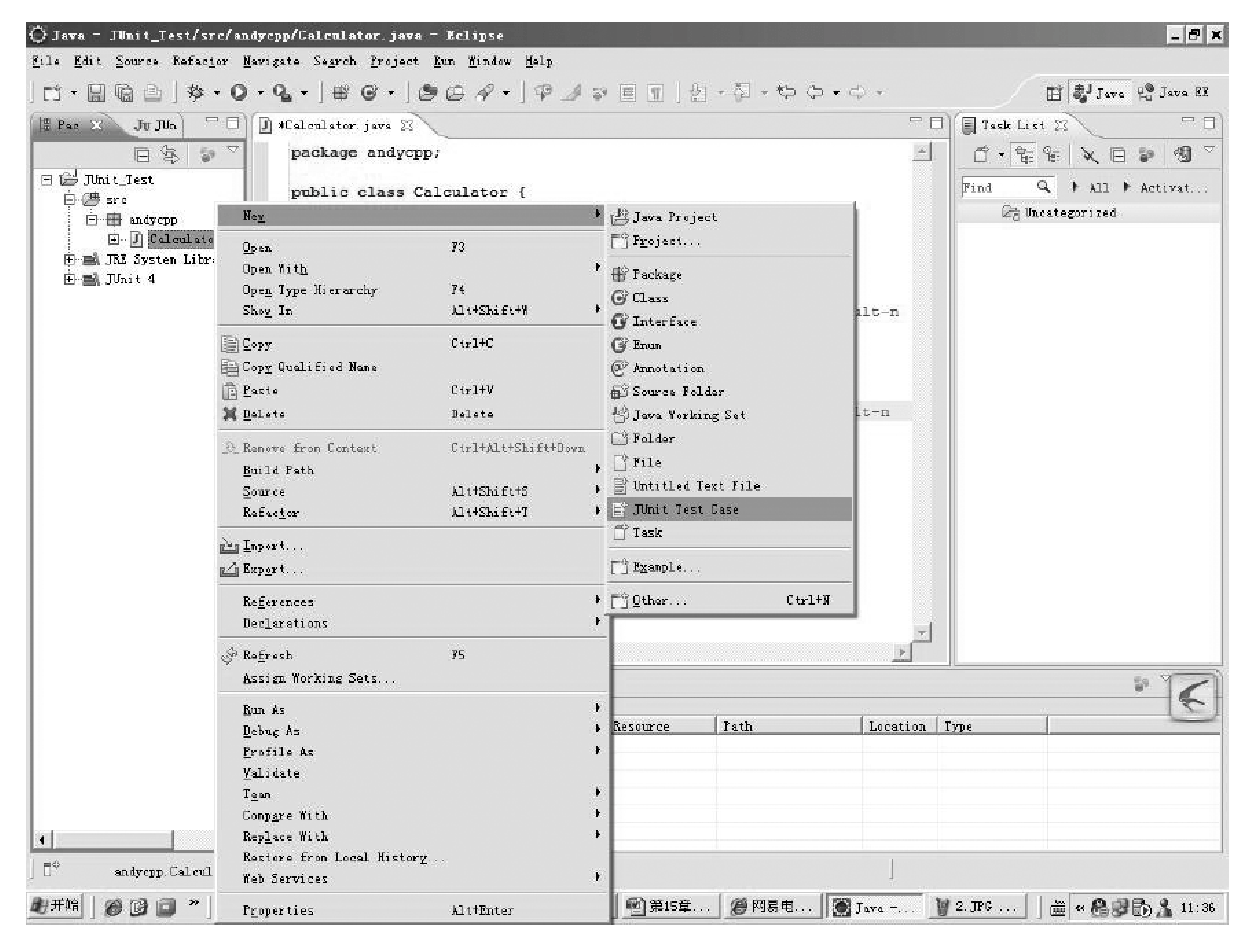


图 3.3　在 Eclipse 中创建 Calculator 类的测试用例截图 1

在弹出的对话框中选择 setUp()和 tearDown()方法,如图 3.4 所示。

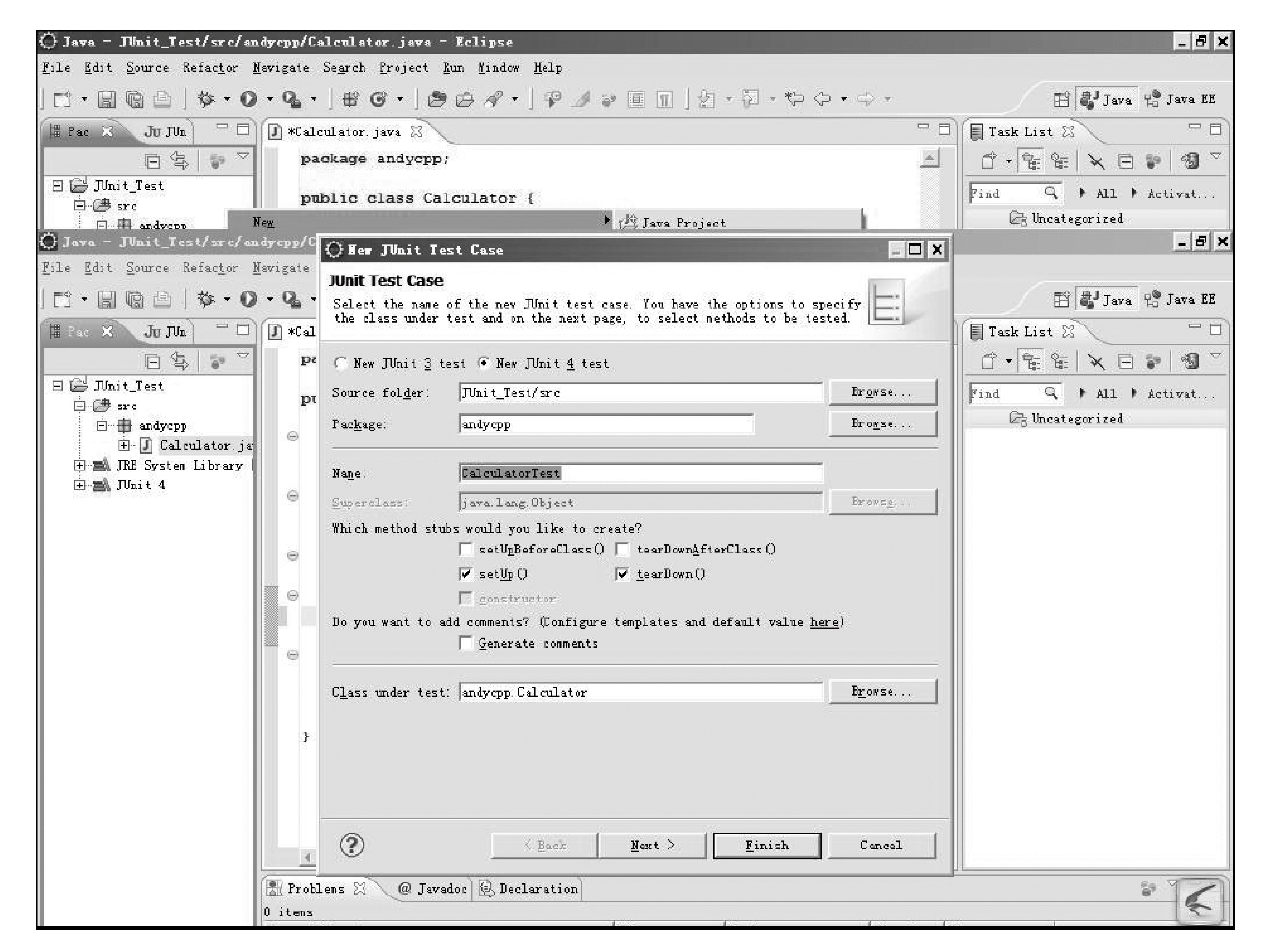


图 3.4　在 Eclipse 中创建 Calculator 类的测试用例截图 2

单击 Next 按钮,系统会自动列出 Calculator 类中所包含的方法,如图 3.5 所示,选择要测试的加、减、乘、除 4 个方法。

Eclipse 自动生成名为 CalculatorTest 的新类,代码中只包含空的测试用例,添加相关的测试用例。

CalculatorTest 类代码如下:

```
package andycpp;
import static org.JUnit.Assert.* ;
import org.JUnit.Before;
import org.JUnit.Ignore;
import org.JUnit.Test;
public class CalculatorTest {
    private static Calculator calculator=new Calculator();
    @Before
    public void setUp()throws Exception {
        calculator.clear();
    }
    @Test
    public void testAdd(){
        calculator.add(2);
        calculator.add(3);
```

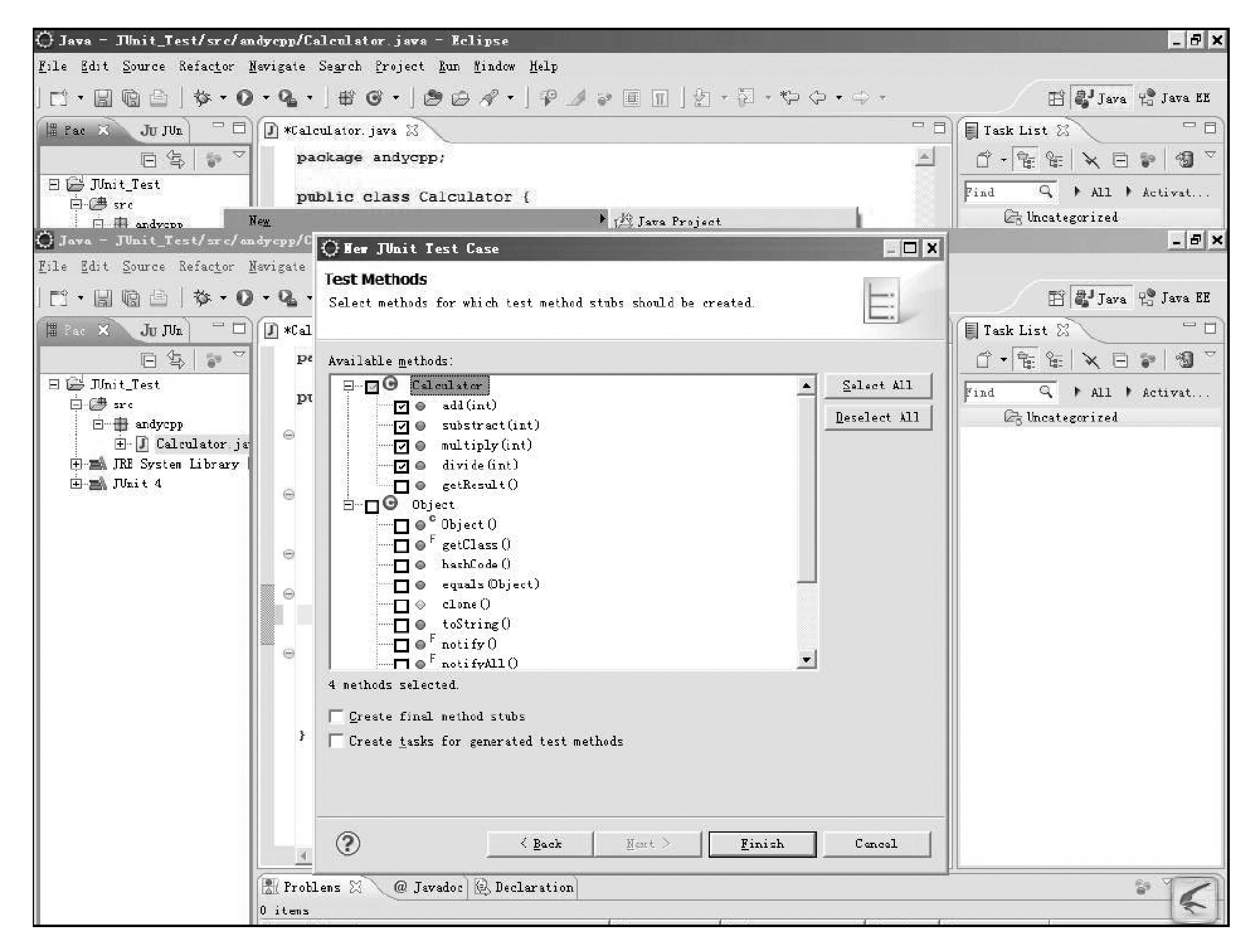


图 3.5　在 Eclipse 中创建 Calculator 类的测试用例截图 3

```
        assertEquals(5, calculator.getResult());
        //用 assertEquals 断言比较执行结果 result 和预期值 5,以判断测试是否成功
    }
    @Test
    public void testSubstract(){
        calculator.add(10);
        calculator.substract(3);
        assertEquals(7, calculator.getResult());
    }
    @Ignore("Multiply()Not yet implemented")
    @Test
    public void testMultiply(){
    }
    @Test
    public void testDivide(){
        calculator.add(6);
        calculator.divide(2);
        assertEquals(3, calculator.getResult());
    }
}
```

步骤 3：运行测试代码。右击 CalculatorTest 类，选择 Run As→Run on Server 命令，如图 3.6 所示。

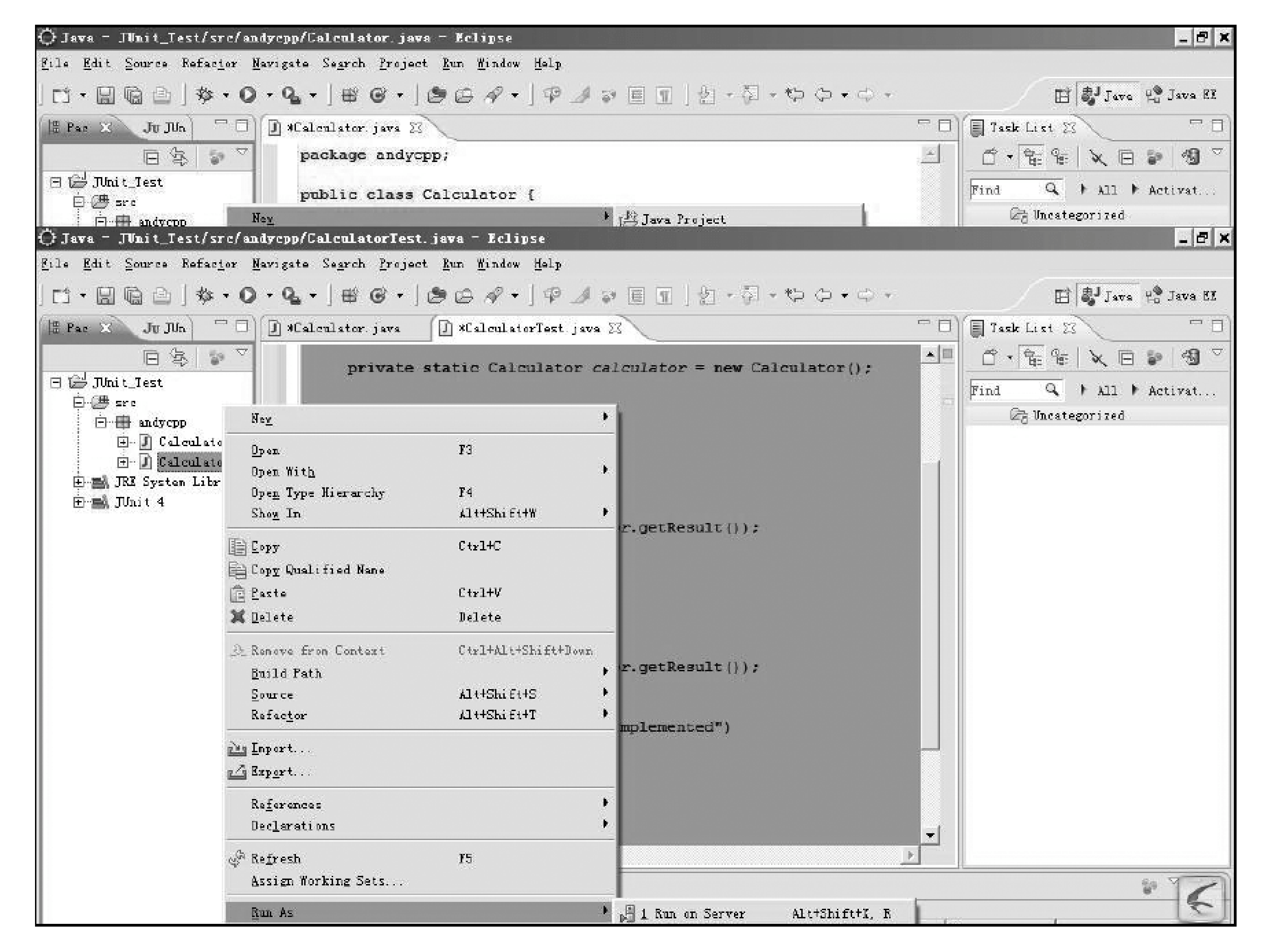


图 3.6　运行测试用例截图 1

运行结果如图 3.7 所示，共进行了 4 个测试，其中 1 个测试被忽略，2 个测试失败。

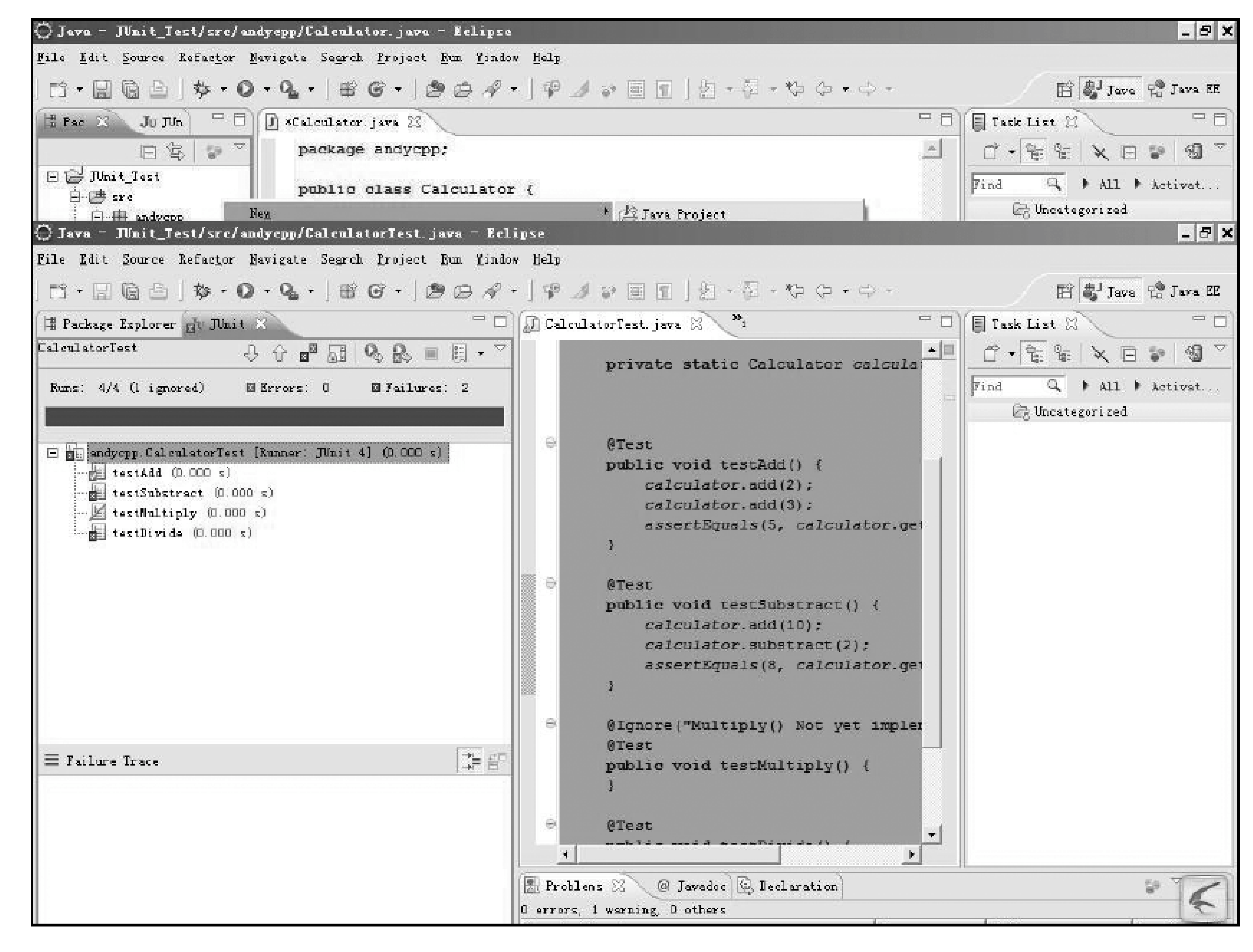


图 3.7　运行测试用例截图 2

修改 Calculator 类代码，重新进行 JUnit 测试，出现绿色的进度条，说明修改正确。

3.3 测试 Sorting 类

3.3.1 Sorting 类

Sorting 类实现了插入排序法、快速排序法、交换等方法。

Sorting 类代码如下：

```
public final class Sorting
{
    public void insertionSort(int[] a)
    {
        int j;
        for(int p=1; p<a.length; p++)
        {
            int tmp=a[p];
            for(j=p; j>0 && tmp<a[j-1]; j--)
                a[j]=a[j-1];
            a[j]=tmp;
        }
    }
    private static void insertionSort(int[] a, int left, int right)
    {
        for(int p=left+1; p <=right; p++)
        {
            int tmp=a[p];
            int j;
            for(j=p; j>left && tmp<a[j-1]; j--)
                a[j]=a[j-1];
            a[j]=tmp;
        }
    }
    public boolean isSorted(int[] a){
        for(int i=0; i<a.length-1; i++){
            if(a[i]>a[i+1]){
                return false;
            }
        }
        return true;
    }
    public static void quicksort(int[] a)
    {
        quicksort(a, 0, a.length-1);
```

```
}
private static final int CUTOFF=10;
private static void quicksort(int[] a, int left, int right)
    {
        //数组元素个数较少时用插入排序,数组元素个数较多时用快速排序
        if(left+CUTOFF <=right)
        {
            int pivot=median3(a, left, right);
            int i=left, j=right-1;
            for(; ;)
            {
                while(a[++i]<pivot)  { }
                while(a[--j]>pivot){ }
                if(i<j)
                    swap(a, i, j);
                else
                    break;
            }
            swap(a, i, right-1);
            quicksort(a, left, i-1);      //对小元素排序
            quicksort(a, i+1, right);     //对大元素排序
        }
        else
            insertionSort(a, left, right);
}
private static int median3(int[] a, int left, int right)
    {
        int center=(left+right)/ 2;
        if(a[center]<a[left])
            swap(a, left, center);
        if(a[right]<a[left])
            swap(a, left, right);
        if(a[right]<a[center])
            swap(a, center, right);
        swap(a, center, right-1);
        return a[right-1];
    }
    public static final void swapReferences(Object [] a, int index1, int index2)
                                                                //交换
    {
        Object tmp=a[index1];
        a[index1]=a[index2];
        a[index2]=tmp;
    }
```

```
    public static final void swap(int[] a,int index1,int index2){
        int tmp=a[index1];
        a[index1]=a[index2];
        a[index2]=tmp;
    }
}
```

3.3.2 SortingTest 类

步骤 1：将 JUnit 4 单元测试包引入 JUnit Test 项目。在该项目上右击，选择“属性”命令，在弹出的属性对话框中，首先在左边选择 Java Build Path，然后在右边选择 Libraries 标签，单击 Add Library 按钮，选择 JUnit 4 并单击 Finish 按钮，添加成功后如图 3.8 所示。

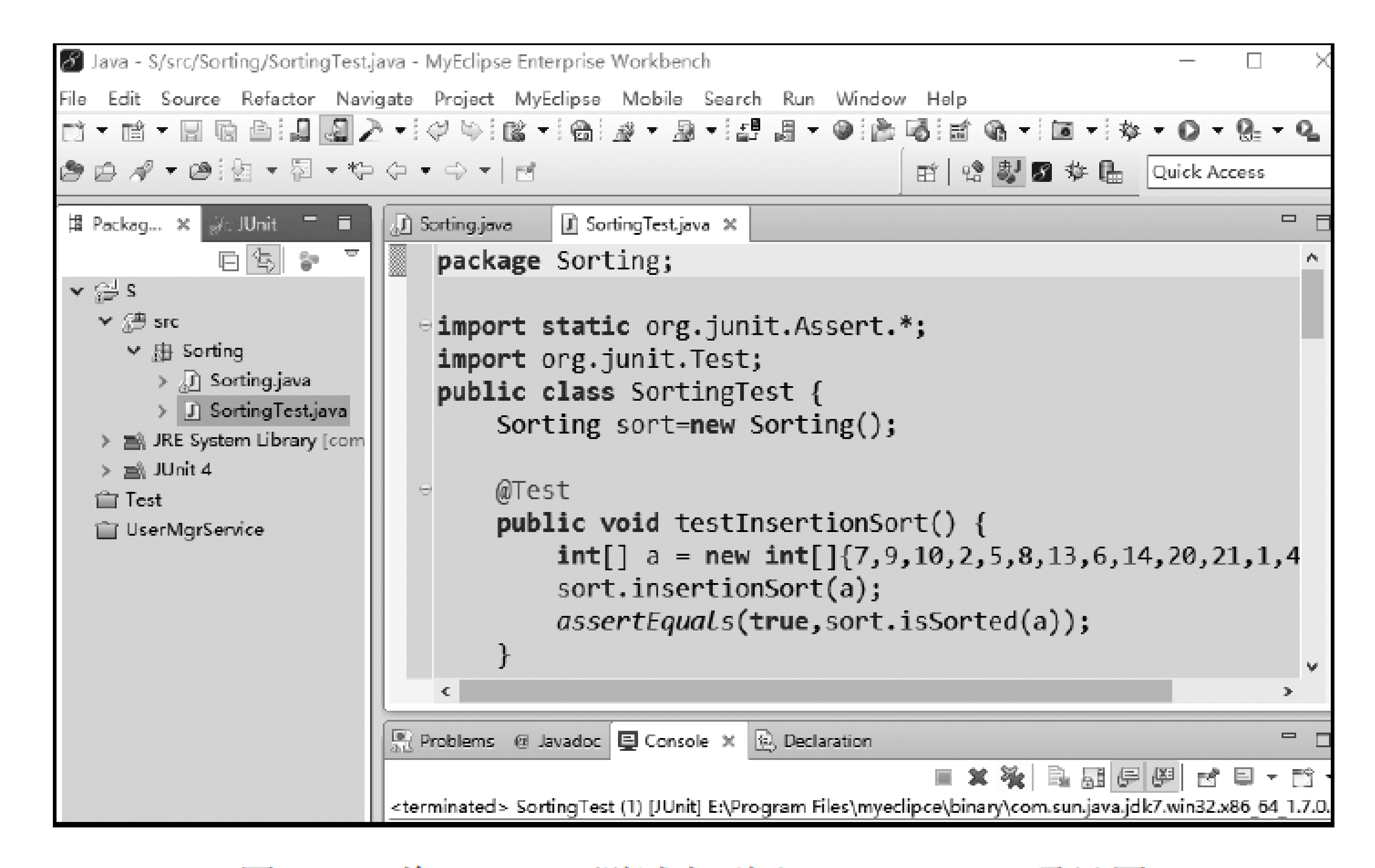


图 3.8 将 JUnit 4 测试包引入 JUnit_Test 项目图

步骤 2：生成 JUnit 测试框架。在 Eclipse 的 Package Explorer 中右击 Sorting 类，选择 New→JUnit Test Case 命令，创建一个 Sorting 类的测试用例，如图 3.9 所示。

步骤 3：单击运行按钮，运行 Sorting 类的测试类，运行结果如图 3.10 所示。

SortingTest 类的代码如下：

```
import static org.JUnit.Assert.*;
import org.JUnit.Test;
public class SortingTest {
    Sorting sort=new Sorting();
    @Test
    public void testInsertionSort(){
        int[] a=new int[]{7,9,10,2,5,8,13,6,14,20,21,1,4,14,24};
        sort.insertionSort(a);
        assertEquals(true,sort.isSorted(a));
    }
```

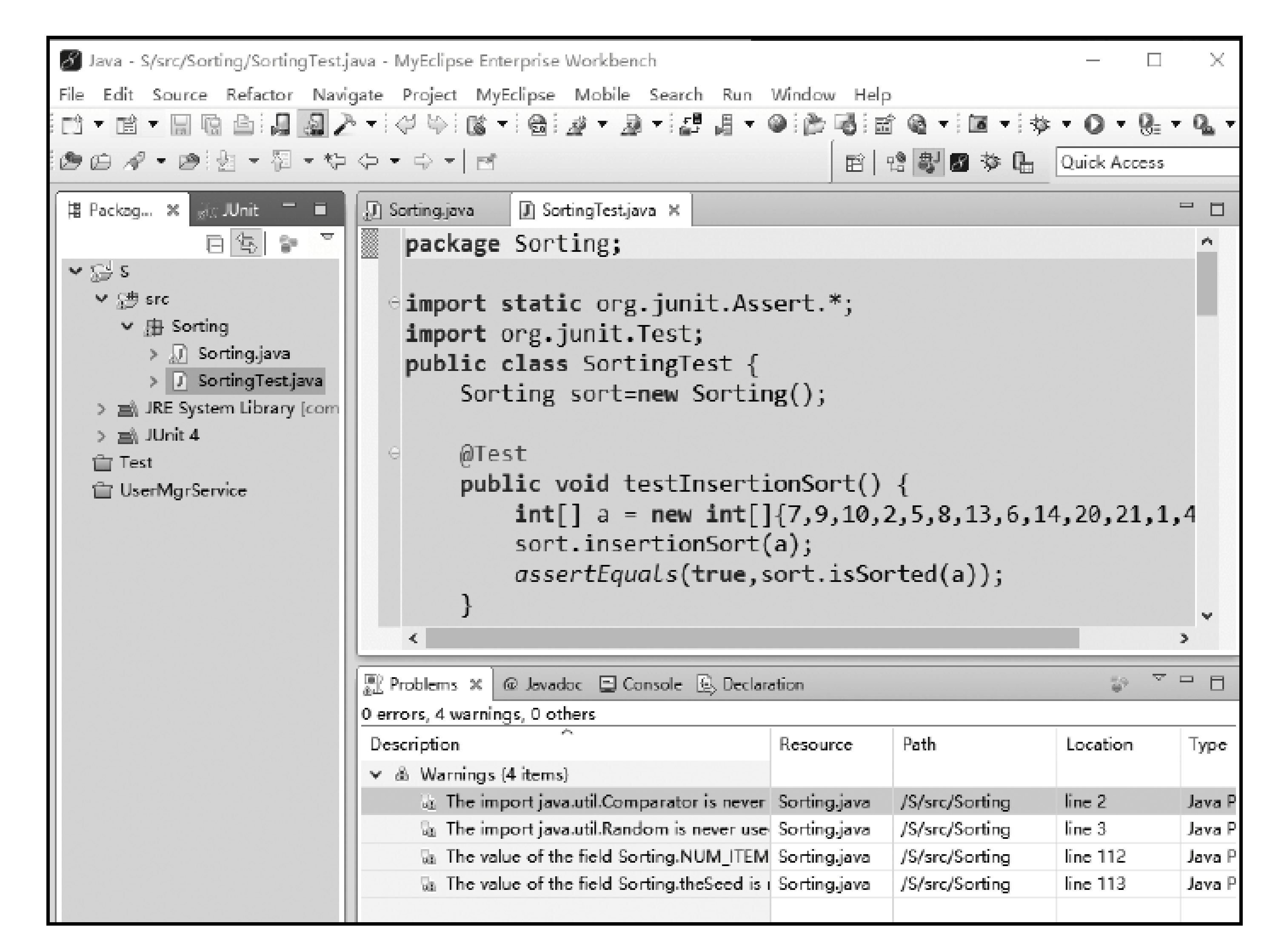


图 3.9 创建一个 Sorting 类的测试类截图

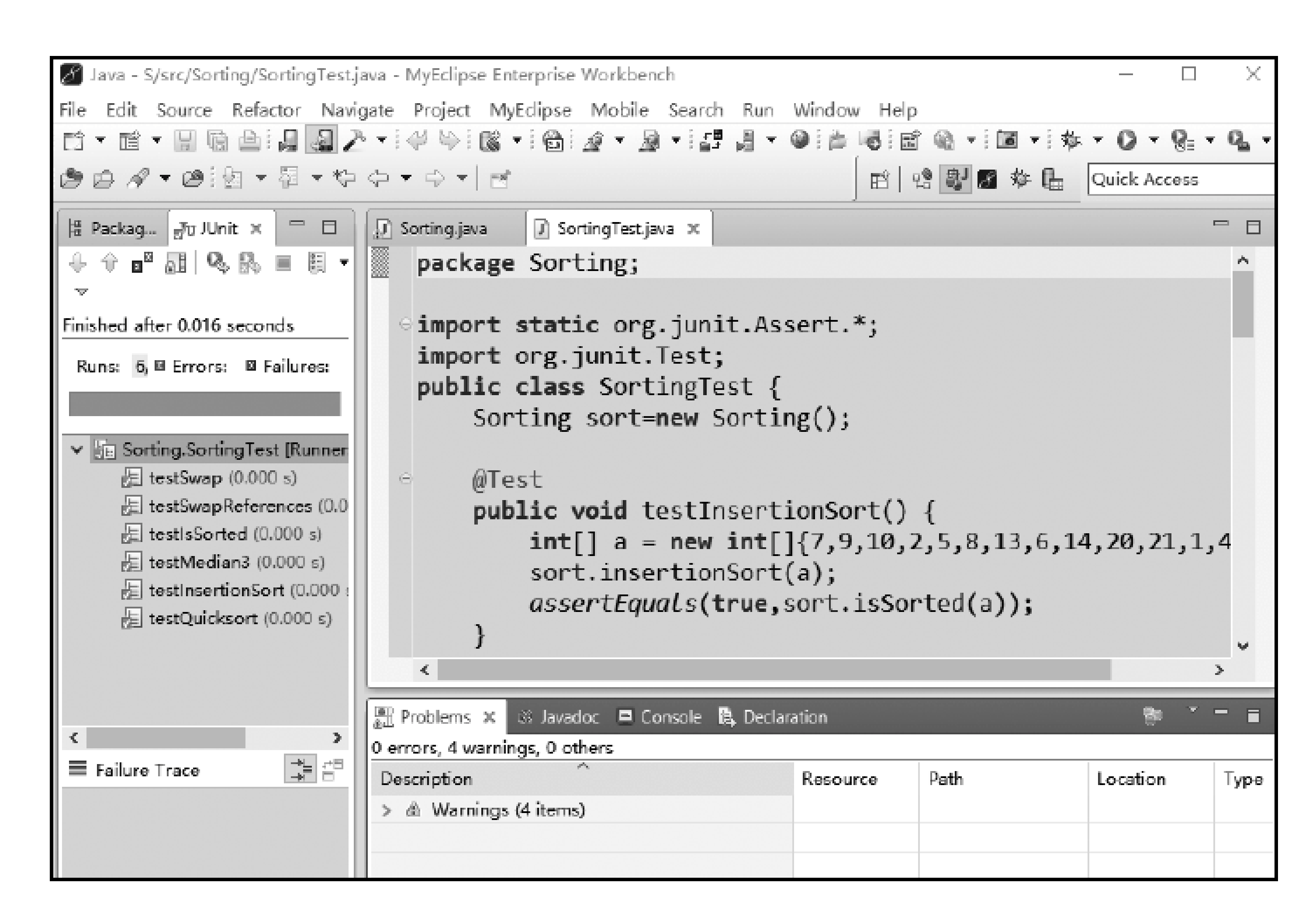


图 3.10 运行 Sorting 类的测试类运行结果图

```
@Test
public void testIsSorted(){
    int[] a=new int[]{7,9,10,2,5,8,13,6,14,20,21,1,4,14,24};
    sort.insertionSort(a);
    assertEquals(true,sort.isSorted(a));
    //交换位置
    int temp=a[4];
    a[4]=a[5];
    a[5]=temp;
    assertEquals(false,sort.isSorted(a));
```

```
        }
        @Test
        public void testQuicksort(){
            int[] a=new int[]{7,9,10,2,5,8,13,6,14,20,21,1,4,14,24};
            Sorting.quicksort(a);
            assertEquals(true,sort.isSorted(a));
            int[] b=new int[]{7,9,10,2,5};
            Sorting.quicksort(b);
            assertEquals(true,sort.isSorted(b));
        }
        @Test
        public void testMedian3(){
            int[] a=new int[]{6,9,10,2,5,8,13,7,14,20,21,1,4,14,24};
            int[] b=new int[]{6,9,10,2,5,8,13,24,14,20,21,1,4,14,7};
            int[] c=new int[]{7,9,10,2,5,8,13,6,14,20,21,1,4,14,24};
            int[] d=new int[]{7,9,10,2,5,8,13,24,14,20,21,1,4,14,6};
            int[] e=new int[]{24,9,10,2,5,8,13,7,14,20,21,1,4,14,6};
            int[] f=new int[]{24,9,10,2,5,8,13,6,14,20,21,1,4,14,7};
            Sorting.quicksort(a);
            Sorting.quicksort(b);
            Sorting.quicksort(c);
            Sorting.quicksort(d);
            Sorting.quicksort(e);
            Sorting.quicksort(f);
            assertEquals(true,sort.isSorted(a));
            assertEquals(true,sort.isSorted(b));
            assertEquals(true,sort.isSorted(c));
            assertEquals(true,sort.isSorted(d));
            assertEquals(true,sort.isSorted(e));
            assertEquals(true,sort.isSorted(f));
            }
    }
```

分析：绿色的进度条说明测试运行通过了。单元测试代码不是用来证明被测代码是正确的，而是为了发现代码错误。因此，单元测试的范围要全面，比如对边界值/正常值/错误值等进行测试。

3.4 测试 WordDealUtil 类

3.4.1 WordDealUtil 类

WordDealUtil 类对名称、地址等字符串格式的内容进行格式检查，将 Java 对象名称(每个单词的头字母大写)按照数据库命名的习惯格式化为小写字母，并且使用下画线分

隔名称中的单词。例如,employeeInfo 经过格式化之后变为 employee_Info。

WordDealUtil 类代码如下:

```
package divide_letter;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
    /* java.util.regex是一个用正则表达式规定的模式对字符串进行匹配的类库包。它包括
       两个类:Pattern 和 Matcher。Pattern 是正则表达式经编译后的表现模式。Matcher 是一
       个状态机器,它以 Pattern 对象作为匹配模式对字符串进行匹配检查。 */
public class WordDealUtil {
    public static String wordFormat4DB(String name)
    {
        Pattern p=Pattern.compile("[A-Z]");
                                    //将给定的正则表达式编译并赋予给 Pattern 类
        Matcher m=p.matcher(name);
        StringBuffer sb=new StringBuffer();
        while(m.find()){
            //if()
            m.appendReplacement(sb, "_"+m.group());
            //group()返回当前查找而获得的与组匹配的所有子串内容
            //appendReplacement(StringBuffer sb, String replacement)
            //将当前匹配子串替换为指定字符串,并且将替换后的子串以及上次匹配子串之后
            //的字符串添加到一个 StringBuffer 对象中
        }
        //appendTail 将最后一次匹配后剩余的字符串添加到一个 StringBuffer 对象里
        return m.appendTail(sb).toString().toLowerCase();
    }
}
```

3.4.2 WordDealUtilTest 测试类

WordDealUtilTest 测试类代码如下:

```
package divide_letter;
import static org.junit.Assert.*;
import org.junit.Test;
public class WordDealUtilTest {
/*  @Test
    public void WordFormat4DBNormal(){
        String target="employeeInfo";
        String result=WordDealUtil.wordFormat4DB(target);
        assertEquals("employee_info",result);
    }*/
    @Test
    public void WordFormat4DBNull(){
```

```
    //测试 null 时的处理情况
        String target=null;
        String result=WordDealUtil.wordFormat4DB(target);
        assertNull(result);
    }
    @Test
    public void WordFormat4DBEmpty(){
    //测试空字符串的处理情况
        String target="";
        String result=WordDealUtil.wordFormat4DB(target);
        assertEquals("",result);
    }
    @Test
    public void WordFormat4DBegin(){
    //测试首字母大写时的处理情况
        String target="EmployeeInfo";
        String result=WordDealUtil.wordFormat4DB(target);
        assertEquals("employee_info",result);
    }
    @Test
    public void WordFormat4DBEnd(){
    //测试尾字母大写时的处理情况
        String target="employeeInfoA";
        String result=WordDealUtil.wordFormat4DB(target);
        assertEquals("employee_info_a",result);
    }
    @Test
    public void WordFormat4DTogether(){
    //测试多个相邻字母大写时的处理情况
        String target="employeeAInfo";
        String result=WordDealUtil.wordFormat4DB(target);
        assertEquals("employee_a_info",result);
    }
}
```

JUnit 运行后提示两个测试情况未通过：

(1) 当首字母大写时得到的处理结果与预期有偏差，造成测试失败(failure)。

(2) 对 null 测试的处理结果直接抛出了测试错误(error)。

分析可知，被测代码中并没有对首字母大写和 null 这两种特殊的情况进行处理，如图 3.11 所示。

在 wordFormat4DB 中添加如下代码，用于对首字母大写和 null 这两种特殊的情况进行处理，再次运行 JUnit，程序正确。

```
//对 null 进行判断
```

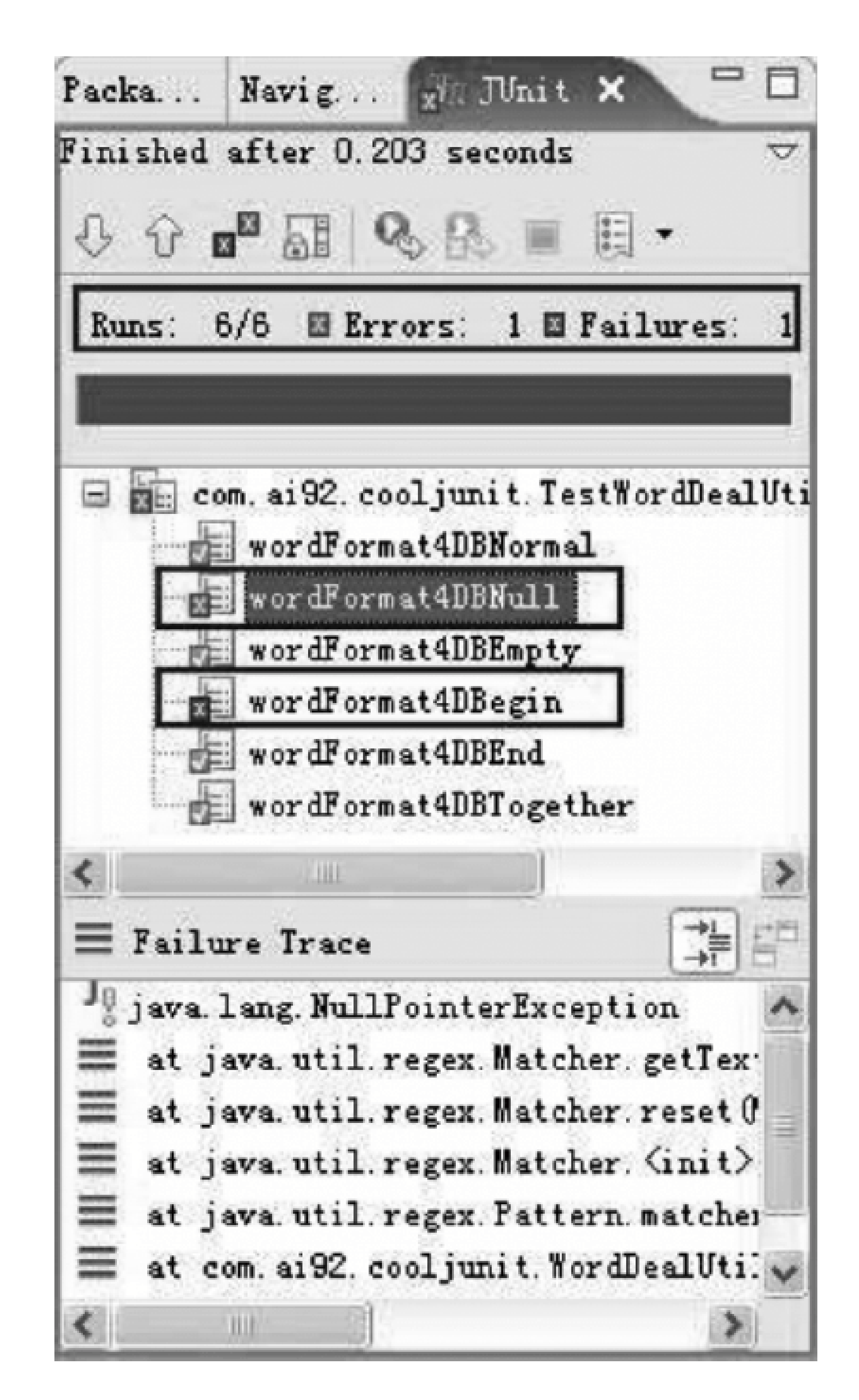


图 3.11　JUnit 测试 WordDealUtilTest 结果图

```
    if(name==null)
        return null;
//对首字母大写的判断
    if(name!=""&&sb.charAt(0)=='_')
        sb=sb.deleteCharAt(0);
```

3.5　测试 Triangle 类

3.5.1　Triangle 类

Triangle 类实现了通过三角形三边对三角形形状的判定，如非三角形(Illegal)、一般三角形(Scalene)、等腰三角形(Isosceles)、等边三角形(Regular)。

Triangle 类代码如下：

```
public class Triangle {
    protected long lborderA=0;
    protected long lborderB=0;
    protected long lborderC=0;
    //构造函数
    public Triangle(long lborderA, long lborderB, long lborderC){
        this.lborderA=lborderA;
```

```
            this.lborderB=lborderB;
            this.lborderC=lborderC;
        }
        //检测是否三角形
        public boolean isTriangle(Triangle triangle){
            boolean isTriangle=false;
        //检测取值范围
            if((triangle.lborderA>0 && triangle.lborderA <=Long.MAX_VALUE)
                    &&(triangle.lborderB>0 && triangle.lborderB <=Long.MAX_VALUE)
                    &&(triangle.lborderC>0 && triangle.lborderC <=Long.MAX_VALUE)){
                //检测是否两边之差小于第三边
                if(diffOfBorders(triangle.lborderA, triangle.lborderB)
                <triangle.lborderC
                    && diffOfBorders(triangle.lborderB, triangle.lborderC)<
                    triangle.lborderA
                    && diffOfBorders(triangle.lborderC, triangle.lborderA)<
                    triangle.lborderB){
                    isTriangle=true;
                }
            }
            return isTriangle;
        }
    //检测三角形类型:非三角形、一般三角形、等腰三角形、等边三角形
        public String getType(Triangle triangle){
            String strType="Illegal";
            if(isTriangle(triangle)){
                //等边三角形
                if(triangle.lborderA ==triangle.lborderB
                        && triangle.lborderB ==triangle.lborderC){
                    strType="Regular";
                }
                //一般三角形
                else if((triangle.lborderA !=triangle.lborderB)
                        &&(triangle.lborderB !=triangle.lborderC)
                        &&(triangle.lborderA !=triangle.lborderC)){
                    strType="Scalene";
                }
                //等腰三角形
                else {
                    strType="Isosceles";
                }
            }
            return strType;
        }
```

```
    //计算两边之差
    public long diffOfBorders(long a, long b){
        return(a>b)?(a-b):(b-a);
    }
//得到边的长度
    public long[] getBorders(){
        long[] borders=new long[3];
        borders[0]=this.lborderA;
        borders[1]=this.lborderB;
        borders[2]=this.lborderC;
        return borders;
    }
}
```

3.5.2 TriangleTest 类

TriangleTest 类代码如下：

```
import static org.junit.Assert.*;
import org.junit.Test;
public class TriangleTest {
    @Test
    public void test1(){
        Triangle triangle=new Triangle(2, 3, 6);
        String actualType=triangle.getType(triangle);
        String expectedType="Illegal";
        assertEquals(expectedType, actualType);
    }
    @Test
    public void test2(){
        Triangle triangle=new Triangle(3, 3, 3);
        String actualType=triangle.getType(triangle);
        String expectedType="Regular";
        assertEquals(expectedType, actualType);
    }
    @Test
    public void test3(){
        Triangle triangle=new Triangle(2, 3, 4);
        String actualType=triangle.getType(triangle);
        String expectedType="Scalene";
        assertEquals(expectedType, actualType);
    }
    @Test
    public void test4(){
        Triangle triangle=new Triangle(3, 3, 4);
```

```
        String actualType=triangle.getType(triangle);
        String expectedType="Isosceles";
        assertEquals(expectedType, actualType);
    }
    @Test
    public void test5(){
        Triangle triangle=new Triangle(-1, 3, 4);
        String actualType=triangle.getType(triangle);
        String expectedType="Illegal";
        assertEquals(expectedType, actualType);
    }
}
```



测试管理软件 TestDirector

实验目的：

(1) 理解 TestDirector 的功能。

(2) 熟练掌握 TestDirector 的操作步骤。

实验环境：TestDirector 软件。

4.1 TestDirector 简介

TestDirector 是 HP 公司推出的基于 Web 的测试管理工具，能够系统地控制整个测试过程，并创建整个测试工作流的框架和基础，使整个测试管理过程变得更为简单和有组织。TestDirector 通过工程数据库，将每一个测试点都对应一个指定的测试需求，并提供直观和有效的方式来计划和执行测试集，收集测试结果并分析数据。

TestDirector 提供完善的缺陷跟踪系统，能够跟踪缺陷从产生到最终解决的全过程。TestDirector 能够与 WinRunner、LoadRunner 等需求和配置管理工具、建模工具无缝链接，提供全套解决方案选择来进行全部自动化的应用测试。

TestDirector 的测试管理包括需求定义、测试计划、测试执行和缺陷跟踪 4 个阶段。

1. 需求定义

需求分析用于分析应用程序并确定测试需求。具体内容如下：

(1) 定义测试范围。检查应用程序文档，并确定测试范围——测试目的、目标和策略。

(2) 创建需求。创建需求树，并确定它涵盖所有的测试需求。

(3) 描述需求。为需求树中的每一个需求主题建立一个详细的目录，并描述每一个需求，给它分配一个优先级，如有必要还可以加上附件。

(4) 分析需求。产生报告和图表以帮助分析测试需求，并检查需求以确保它们在测试范围内。

2. 测试计划

测试计划阶段基于测试需求建立测试计划。具体内容如下：

(1) 定义测试策略。检查应用程序、系统环境和测试资源，并确认测试目标。

(2) 定义测试主题。将应用程序基于模块和功能进行划分,并对应到各个测试单元或主题,构建测试计划树。

(3) 定义测试。定义每个模块的测试类型,并为每一个测试添加基本的说明。

(4) 创建需求覆盖。将每一个测试与测试需求进行连接。

(5) 设计测试步骤。对于每一个测试,先决定其要进行的测试类型,若准备进行手动测试,需要为其在测试计划树上添加相应的测试步骤。测试步骤描述测试的详细操作、检查点和每个测试的预期结果。

(6) 分析测试计划。产生报告和图表来帮助分析测试计划数据,并检查所有测试以确保它们满足测试目标。

3. 测试执行

测试执行创建测试集并执行测试。具体内容如下:

(1) 创建测试集。在工程中定义不同的测试组来达到各种不同的测试目标,并确定每个测试集都包括哪些测试。

(2) 确定进度表。为测试执行制定时间表,并为测试员分配任务。

(3) 运行测试。自动或手动执行每一个测试集。

(4) 分析测试结果。查看测试结果并确保应用程序缺陷已经被发现。生成的报告和图表可以帮助分析这些结果。

4. 缺陷跟踪

缺陷跟踪报告程序中产生的缺陷并跟踪缺陷修复的全过程。具体内容如下:

(1) 添加缺陷。报告程序测试中发现的新的缺陷。在测试过程中的任何阶段,质量保证人员、开发者、项目经理和最终用户都能添加缺陷。

(2) 检查新缺陷。检查新的缺陷,并确定哪些缺陷应该被修复。

(3) 修复打开的缺陷。修复那些决定要修复的缺陷。

(4) 测试新构建。测试应用程序的新构建,重复上面的过程,直到缺陷被修复。

(5) 分析缺陷数据。产生报告和图表来帮助分析缺陷修复过程,并帮助决定什么时候发布该产品。

4.2 TestDirector 操作步骤

1. 启动 TestDirector

在“开始”菜单单击 TestDirector 图标,TestDirector 启动后的首页如图 4.1 所示,地址栏为:http://服务器名(或者 IP 地址)/TDBIN。

单击左侧的 TestDirector,进入 TestDirector 的主界面,选择域和工程,输入用户名和密码,登录进入 TestDirector 主界面。本书选用默认的域和工程,采用自带的 Demo 项目例子,用默认超级用户 Admin,输入密码(默认为空),单击 Login 按钮,登录

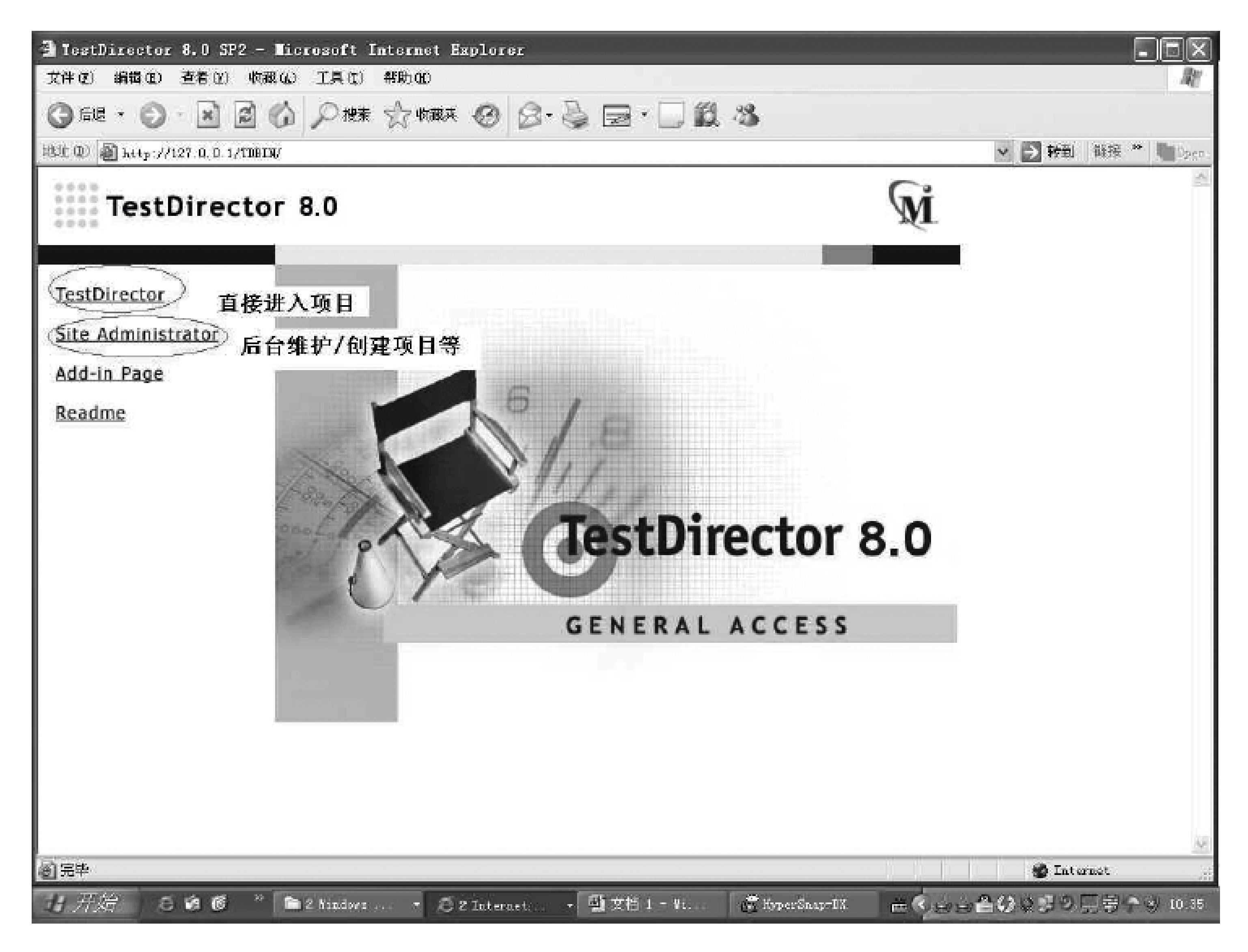


图 4.1　TestDirector 登录界面

TestDirector。

单击 TestDirector 首页左侧的 Site Administrator，输入密码(默认为空)，进入 TestDirector 管理界面，如图 4.2 所示。

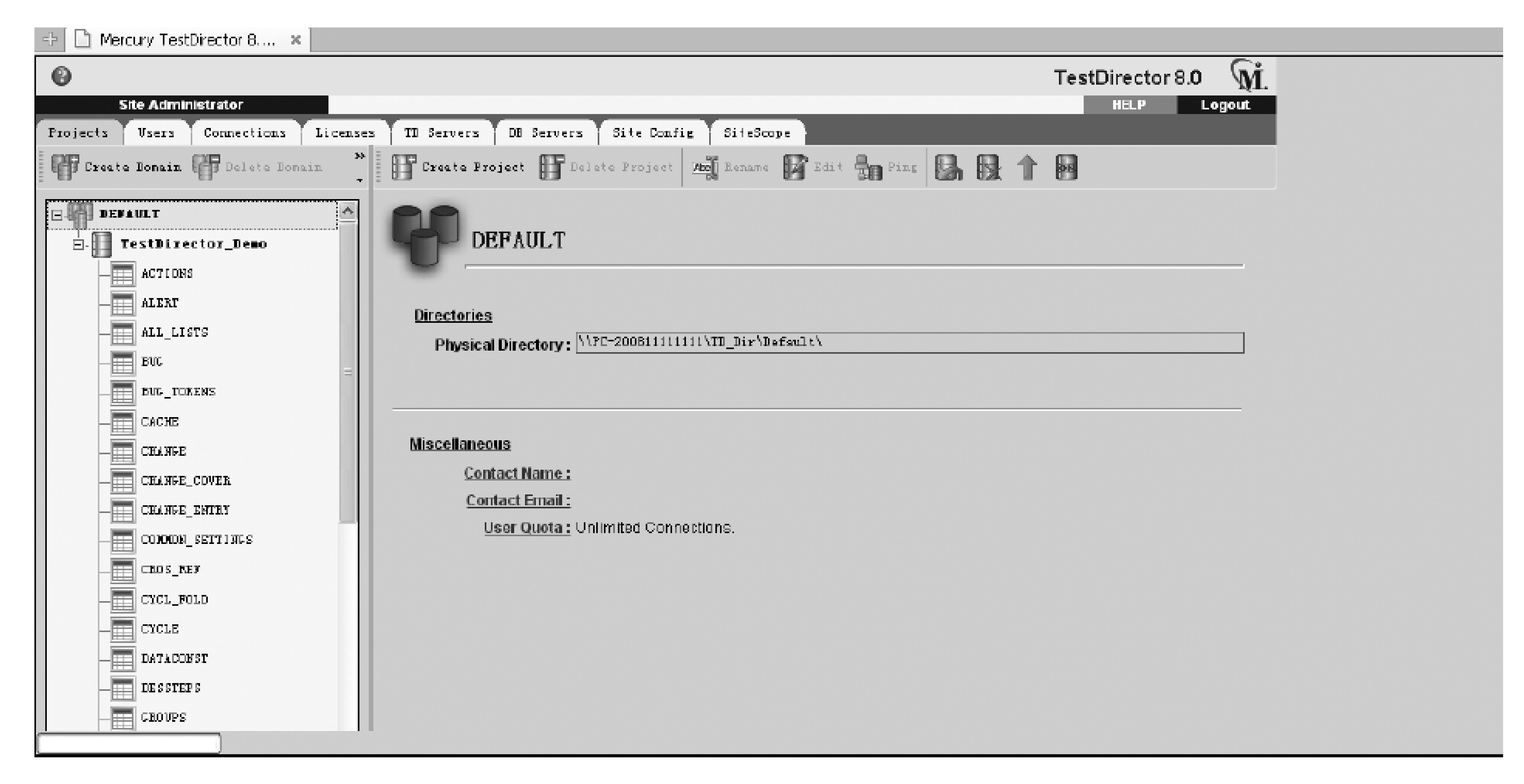


图 4.2　TestDirector Site Administrator 界面

2. TestDirector 主窗口

TestDirector 的主窗口(图 4.3)由以下几部分构成：

- TestDirector Toolbar，显示工程常用的命令。
- Menu Bar，显示模块常用的命令。

• Tools Button，显示常用的工具。



图 4.3 TestDirector 主窗口

3. TestDirector 应用举例

1）项目管理库环境配置

登录 Site Administrator 后，在 Projects 标签页中创建域，只有系统的域用户可以共享这个域的存储区，一个域可以存储多个工程。

当创建一个 TestDirector 工程后，需要存储和管理 TestDirector 自身产生和连接的数据库。每一个工程都支持通过数据库来存储工程信息。在默认域(default)中创建测试项目 tdtest，数据库类型为 Microsoft SQL。在工具栏选择 Create Project，输入项目名称，选择数据库 MS-SQL，如图 4.4 所示。

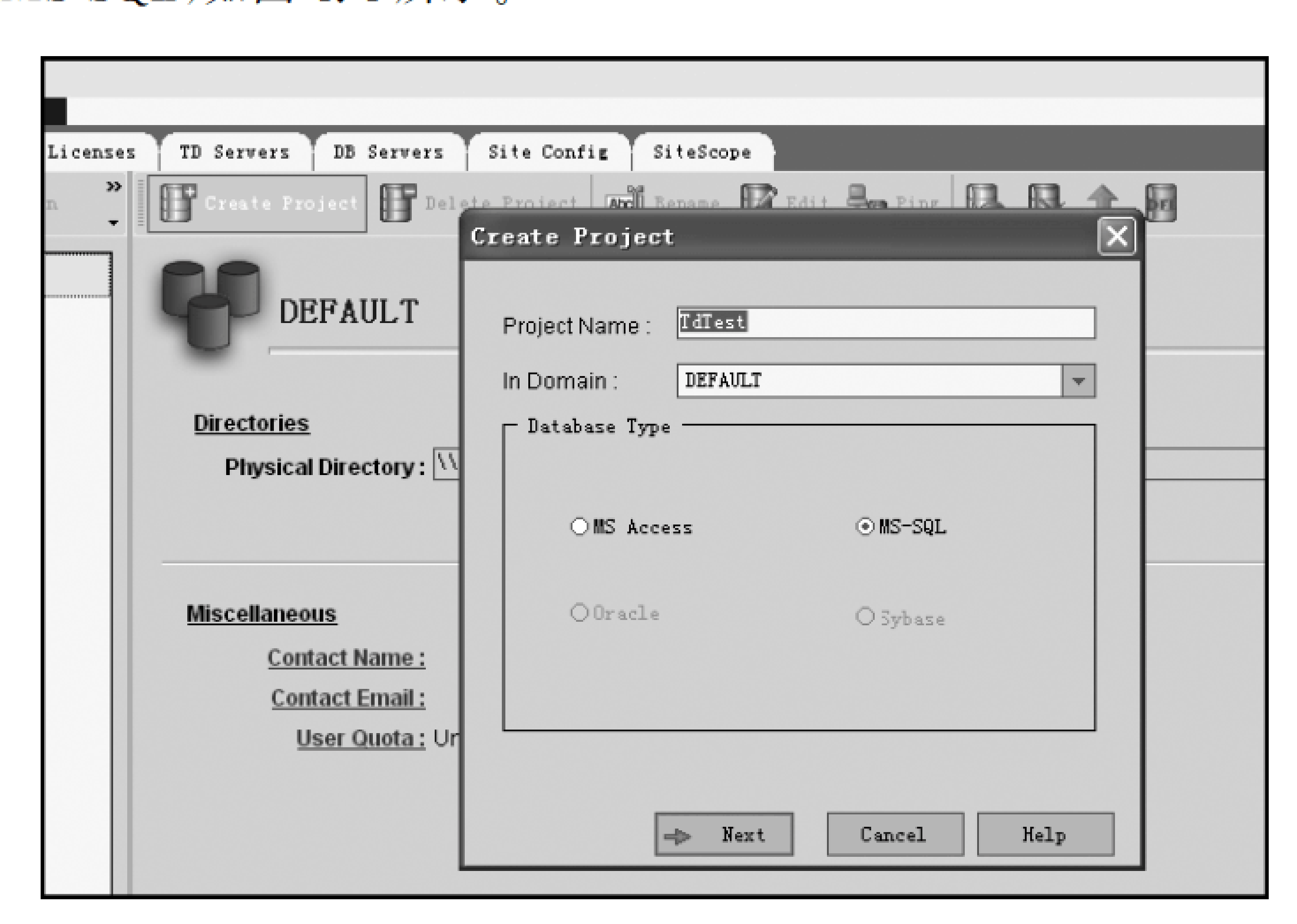


图 4.4 创建测试项目

单击 Next 按钮，输入数据库网络服务器名、数据库用户名和密码(要具有创建数据

库表的权限)，如图 4.5 所示。连接成功后，给出提示信息，如图 4.6 所示。

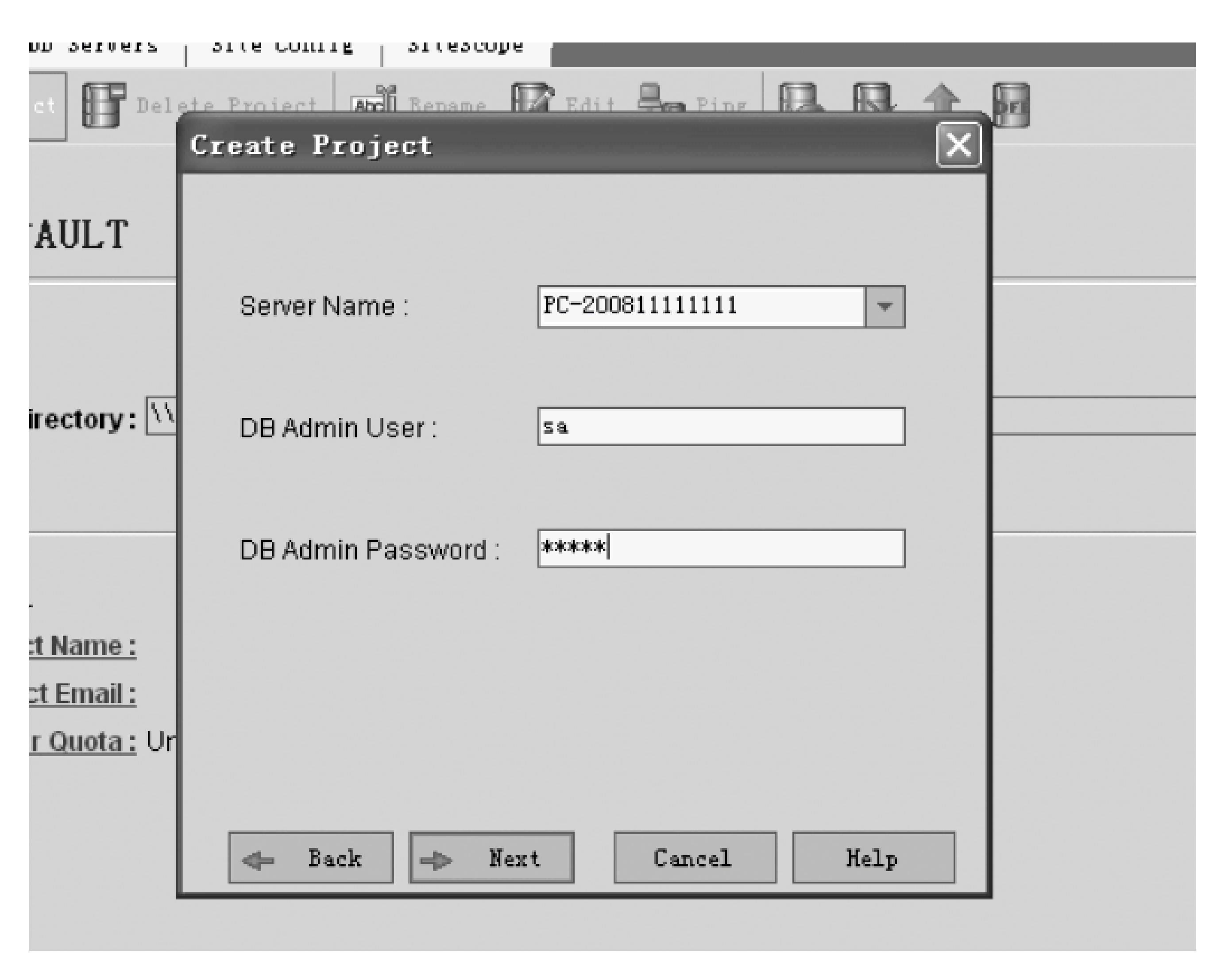


图 4.5　连接数据库

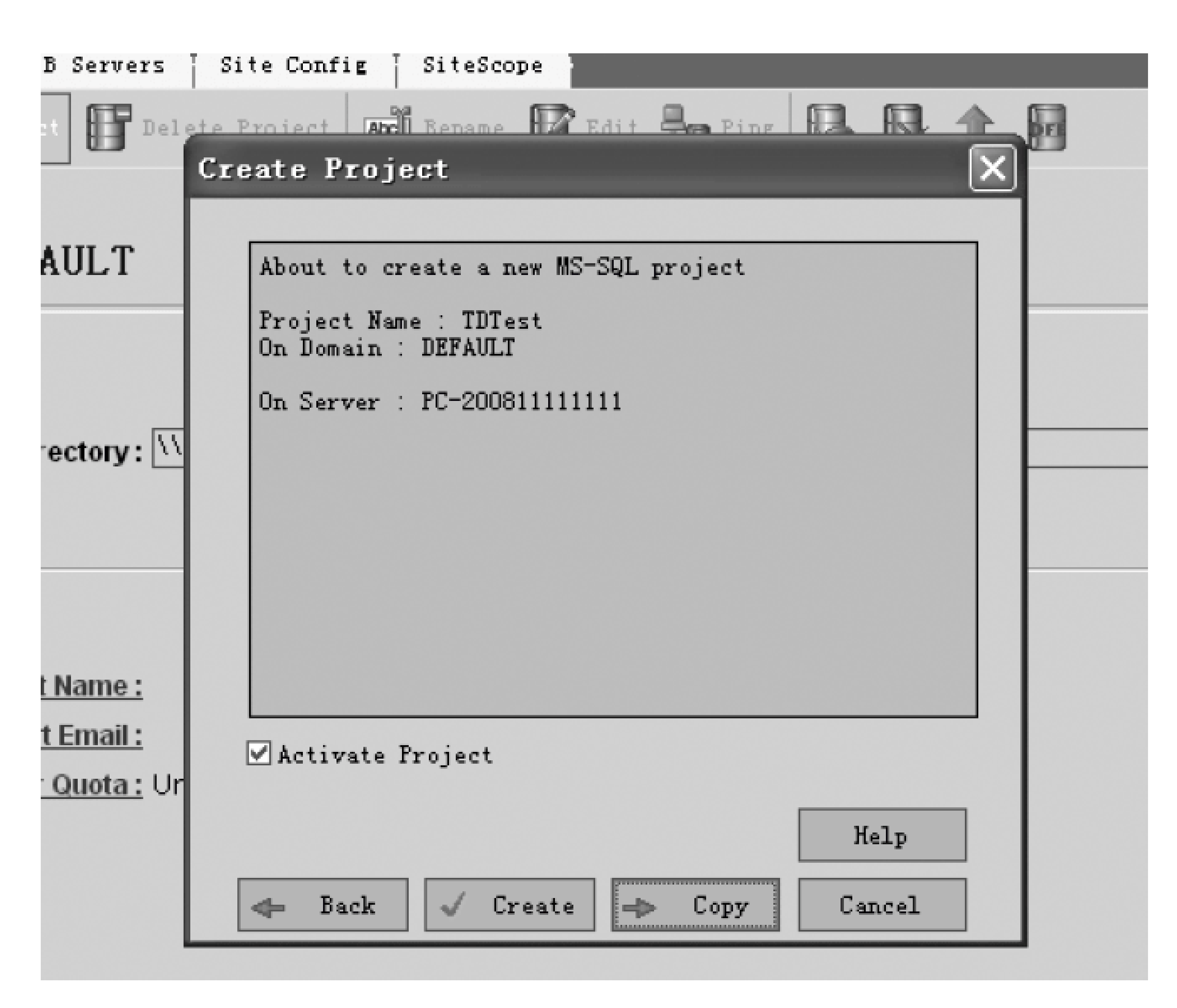


图 4.6　连接成功后的提示信息

单击 Create 按钮，系统创建数据库及相应的表，创建成功后，数据库中自动生成用户，格式为“域名_测试项目名_db”，默认密码为空。对于本例，用户为 default_tdtest_db，密码为空，如图 4.7 所示。

TestDirector 允许管理用户访问工程的权限，它会创建一个拥有权限的用户列表并为一个组或一个用户分配一个口令。在 TestDirector 中用户所拥有的权限是由该用户所在的用户组决定的。TestDirector 允许为工程中指定的目录创建包含特权和许可机制的

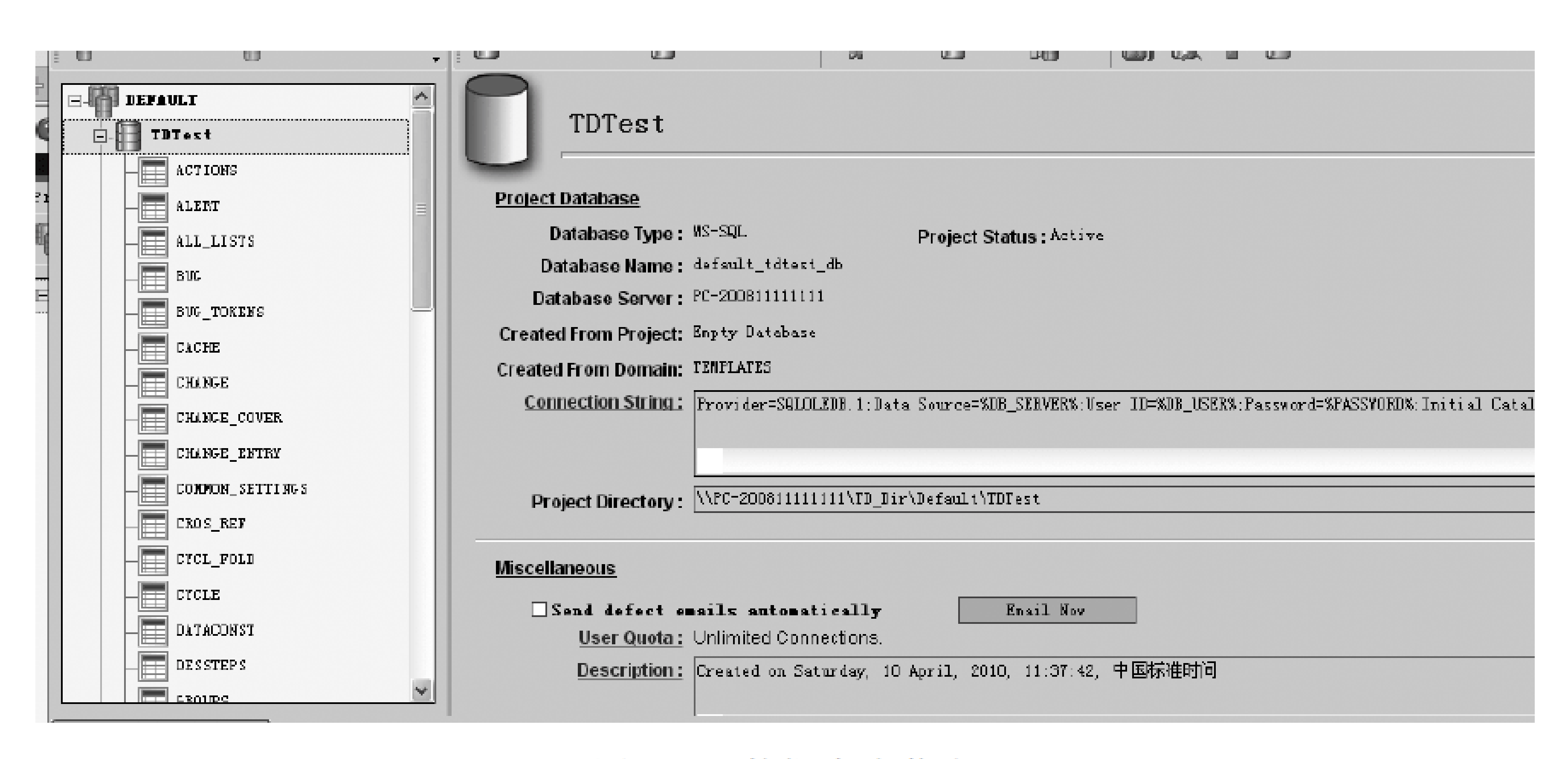


图 4.7 数据库表信息

规则。

在系统中添加用户，单击 Users 标签，选择 New 添加用户，输入新增用户信息，单击 OK 按钮后，成功加入该用户，如图 4.8 所示。

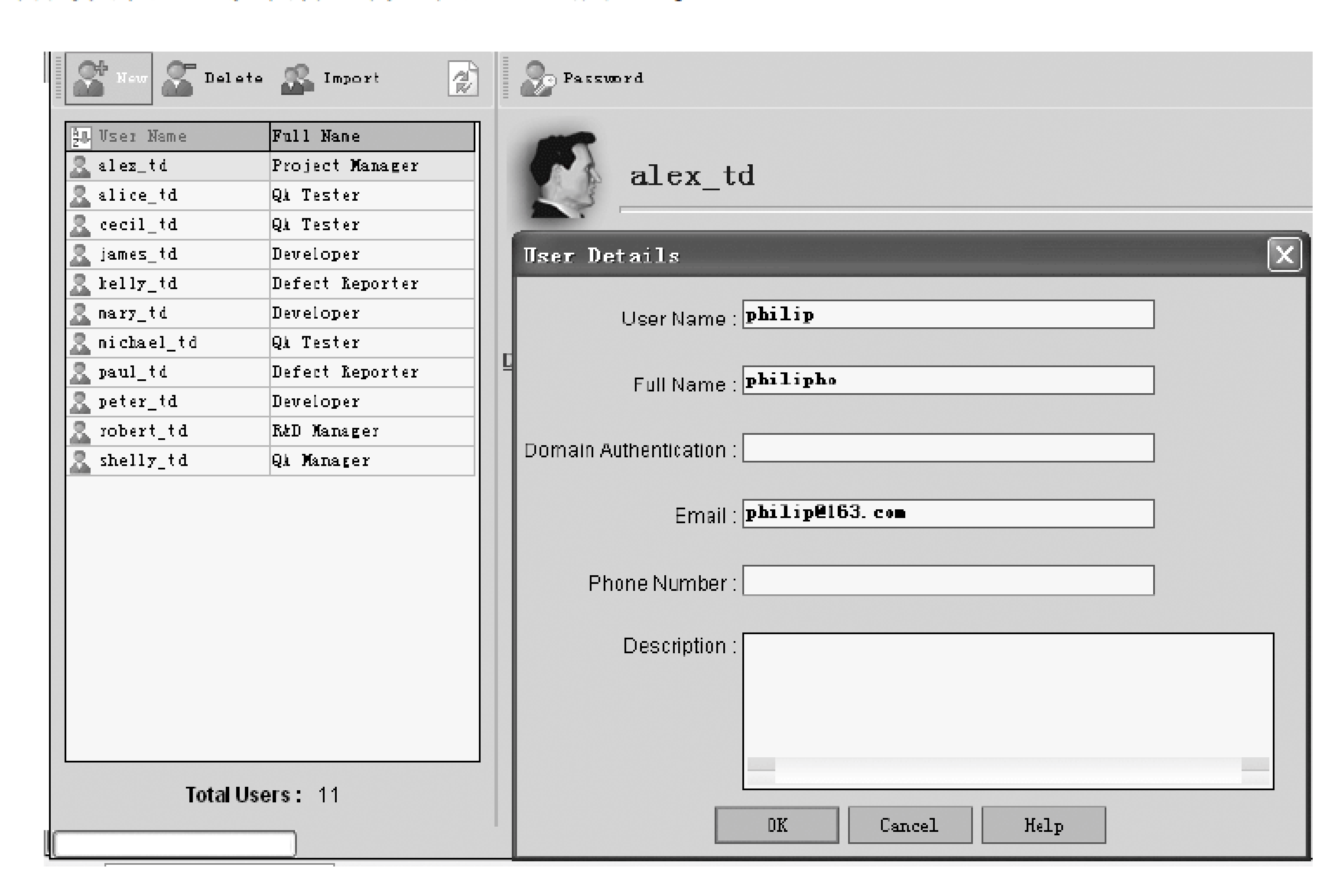


图 4.8 添加用户并输入用户信息

增加项目用户后，退出 Site Administrator 界面，单击 TestDirector 主界面左边链接列表中的 TestDirector 链接进入 TestDirector 登录界面，如图 4.9 所示。在登录界面右上角单击 CUSTOMIZE，为成员分配权限，如图 4.10 所示。

如果某个组(group)拥有了某个权限，那么该组下的成员也都具有该权限。

- TDAdmin：Admin 用户群成员有全部的权限。
- QATester：具有管理需求、测试计划、测试库和缺陷等权限，只能添加和修改缺陷，不能删除缺陷。
- Project Manager：具有管理需求、测试计划、测试库和缺陷的权限。
- Developer：具有管理测试库和缺陷的权限。

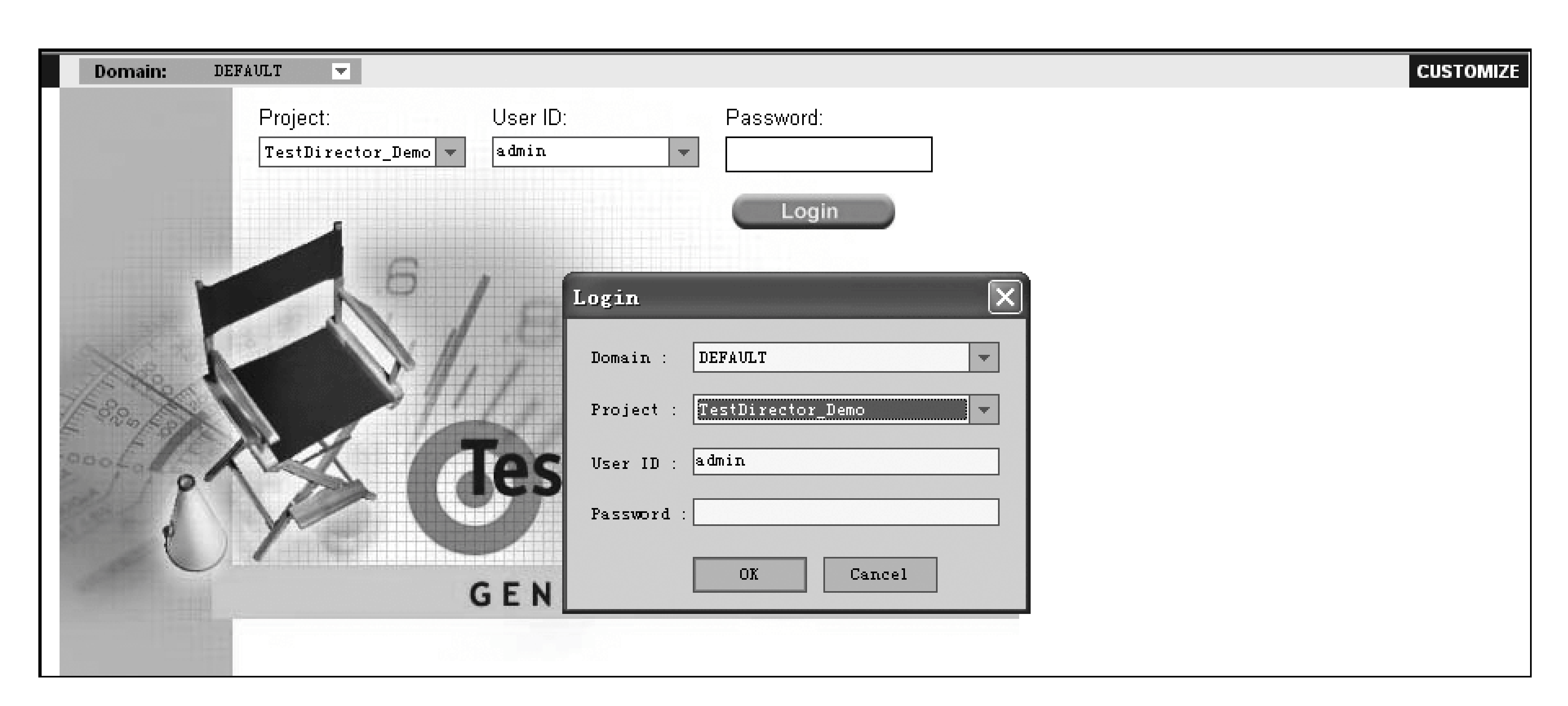


图 4.9 TestDirector 登录界面

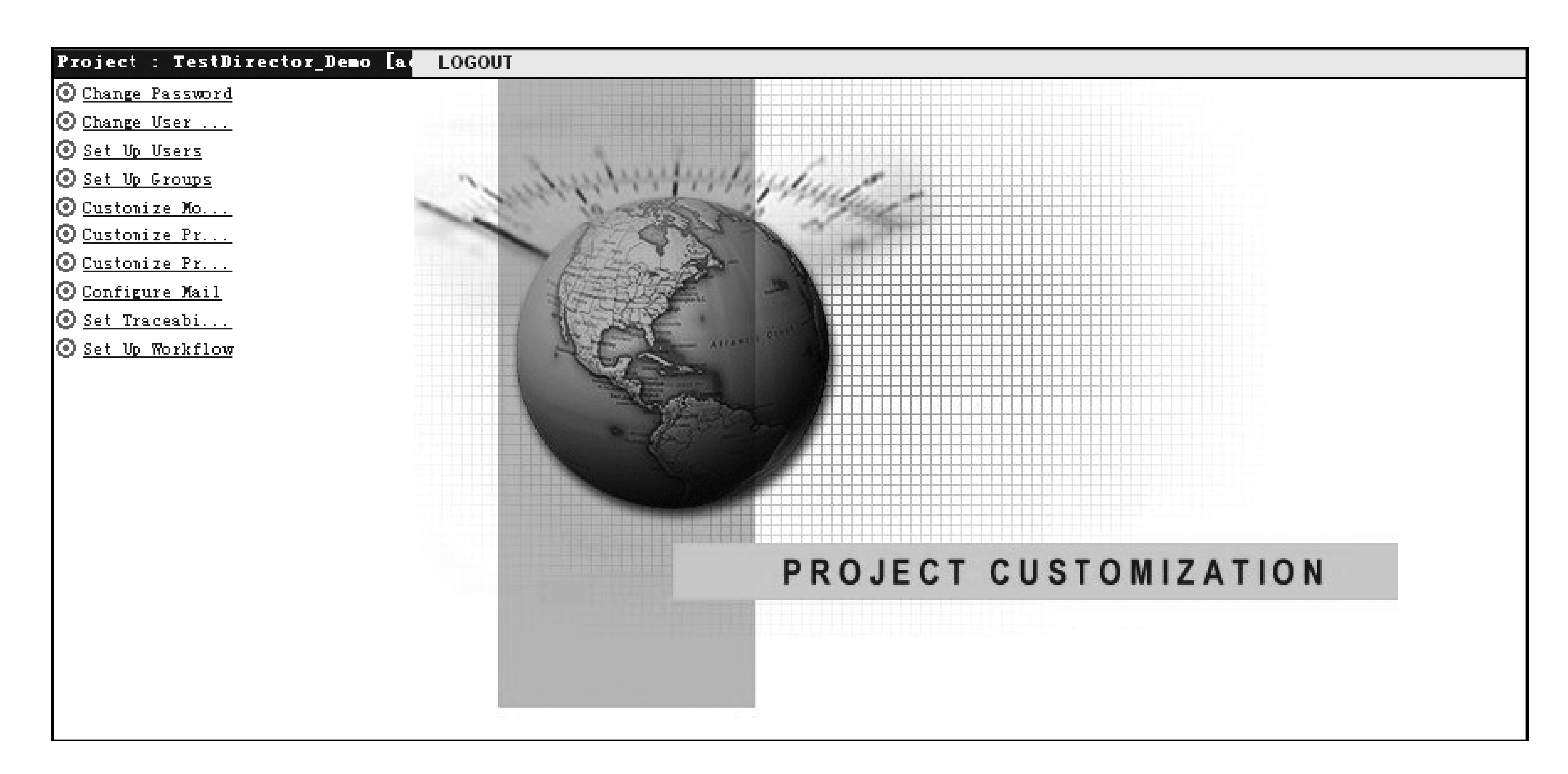


图 4.10 分配权限

- View：在 TestDirector 项目里只有只读权限，只能更改自己的密码和属性。

2）管理测试需求

在执行测试之前，应该指定测试需求。测试需求可以与需求规格说明书的需求一一对应，实际上，有些公司就是用 TestDirector 的需求管理模块来管理和维护需求规格说明书的。

测试需求详细描述了应该测试的范围和内容，给测试组的后续测试过程提供基本依据。测试需求可以与测试的用例和缺陷对应起来，从而实现需求的可跟踪性，并且帮助测试组长和项目经理判断测试的质量以及提供产品满足需求程度的参考。应该通过定义测试需求来开始整个应用程序的测试过程。需求详细地描述了应用程序中哪些需要被测试，并为测试组提供了整个测试过程的基础。通过定义这些需求，能够更好地聚焦于商业需要对测试进行计划和管理。需求与测试和缺陷关联，从而确保整个过程可追溯并帮助整个过程的决策。

在 TestDirector 中通过 REQUIREMENTS 模块的需求树来管理测试需求，如图 4.11 所示。

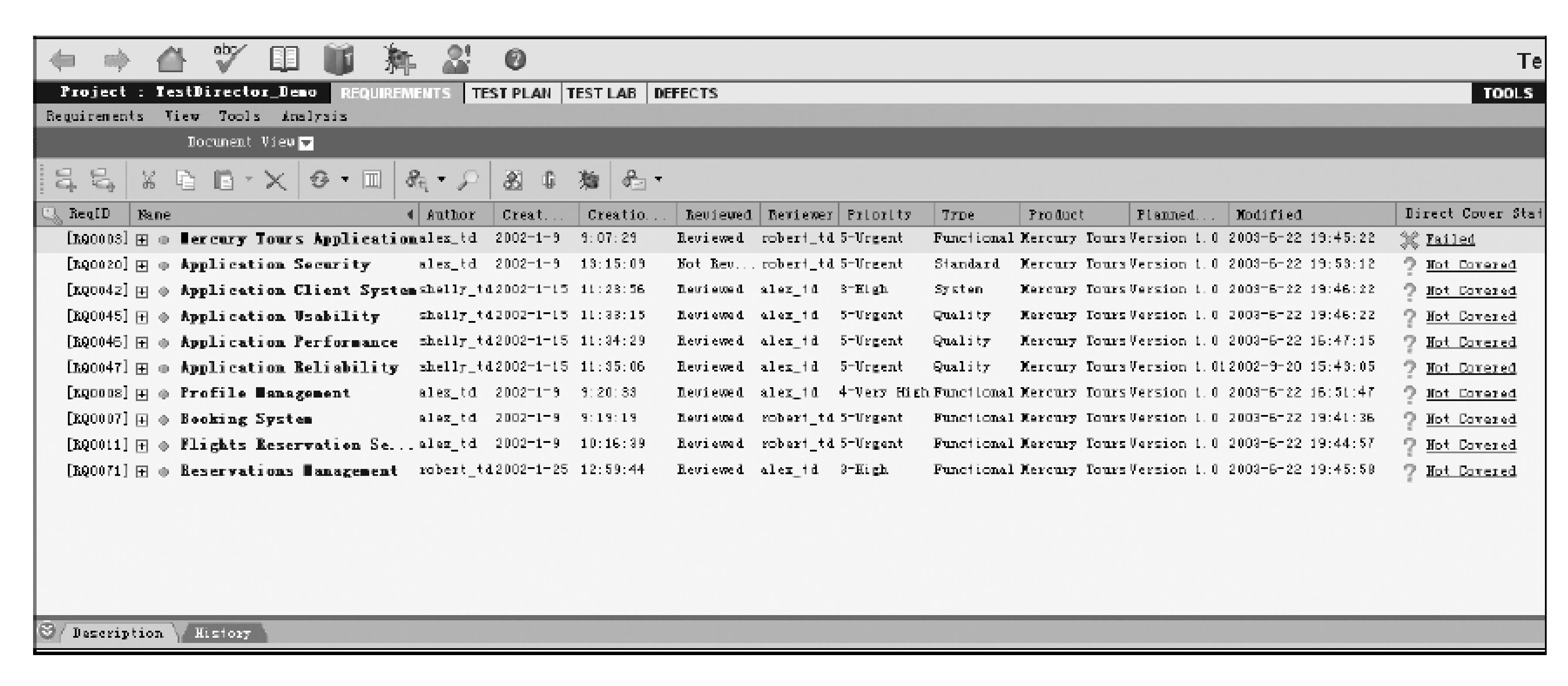


图 4.11　TestDirector 的 REQUIREMENTS 模块

通过选择 Requirements 菜单下的子菜单 New Requirement 和 New Child Requirement 来创建需求项和子需求项，组织需求的层次关系，从而形成需求树。

每一项需求都有唯一的 ReqID 标识。在每一个需求项的 Description 页描述该项需求的具体内容。在 Priority 列中为每一个需求项选择一个优先级，默认优先级划分成以下几个级别：

- 1—Low。
- 2—Medium。
- 3—High。
- 4—Very High。
- 5—Urgent。

可根据需要选择合适的优先级别。测试需求与测试用例可以通过测试覆盖把两者关联起来。关联的操作步骤如下：

(1) 切换到 Coverage View 的覆盖视图。

(2) 选择 Document View 所在的下拉框。

(3) 选中 Coverage View，则以覆盖视图展示需求项及覆盖需求项的测试用例，如图 4.12 所示。

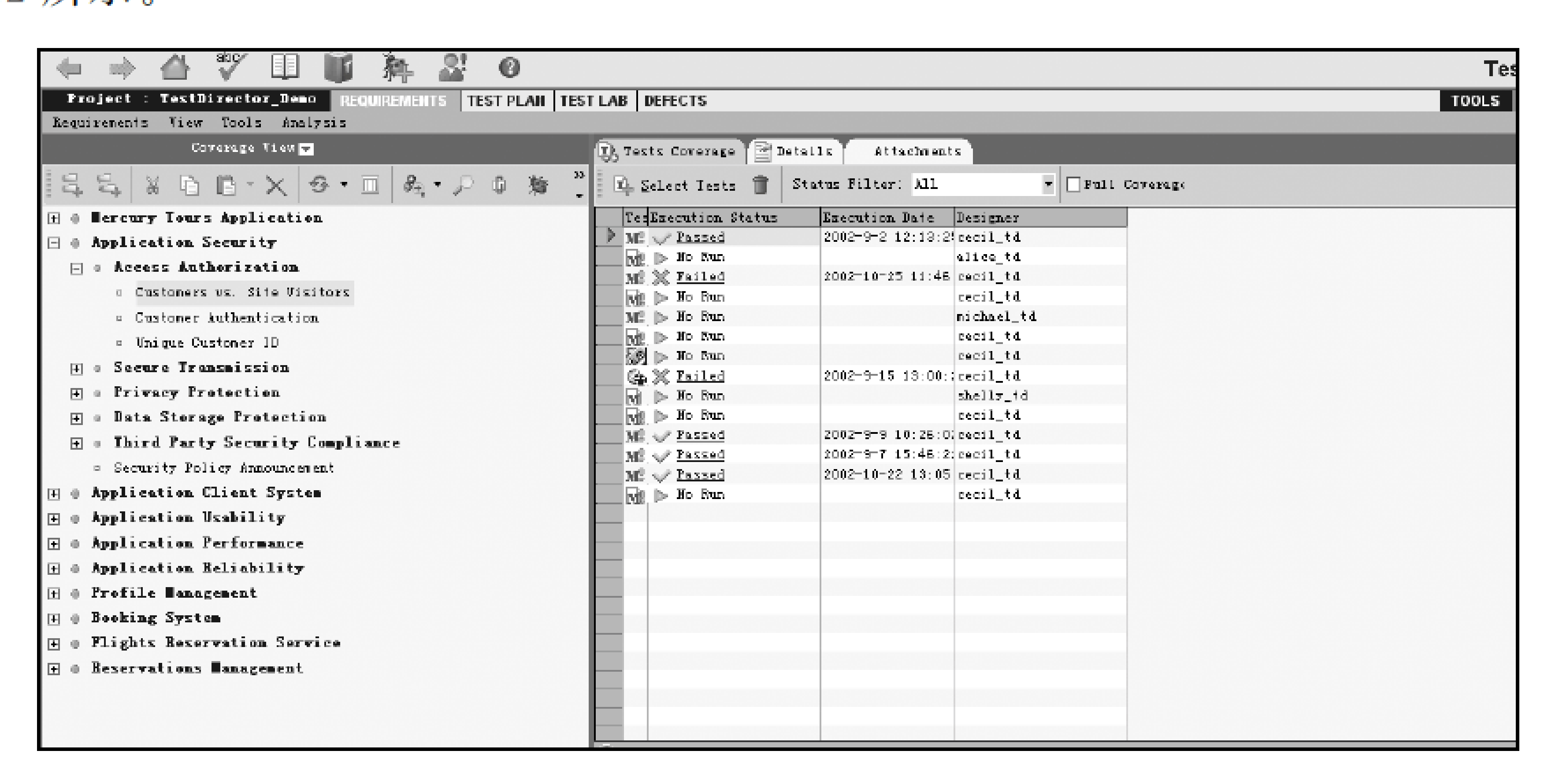


图 4.12　TestDirector 的覆盖视图

在需求树中选中某个需求项后，可在右边的 Tests Coverage 界面单击 Select Tests 按钮，选择需要的测试用例，当然前提是测试用例已经设计好了。这样就把需求和覆盖需求的测试用例关联起来。可以通过计算每一项需求对应的测试用例个数来统计对该项需求的覆盖力度。需求与测试用例之间的关联是相互的，也可以在测试用例设计的界面选择需要覆盖的需求项，来达到关联的目的。这在后面会再讲到。

在测试需求管理模块中，还有生成分析报告的功能，包括测试需求文档报告、需求基本情况报告、需求覆盖情况报告等。图 4.13 展示了需求的优先级分布情况。要生成分析报告，可选择 Analysis 菜单的 Reports 命令，选择需要报告的类型，该项用于产生文本类型的报告；而选择 Graphs 命令则用于产生图表类型的报告。

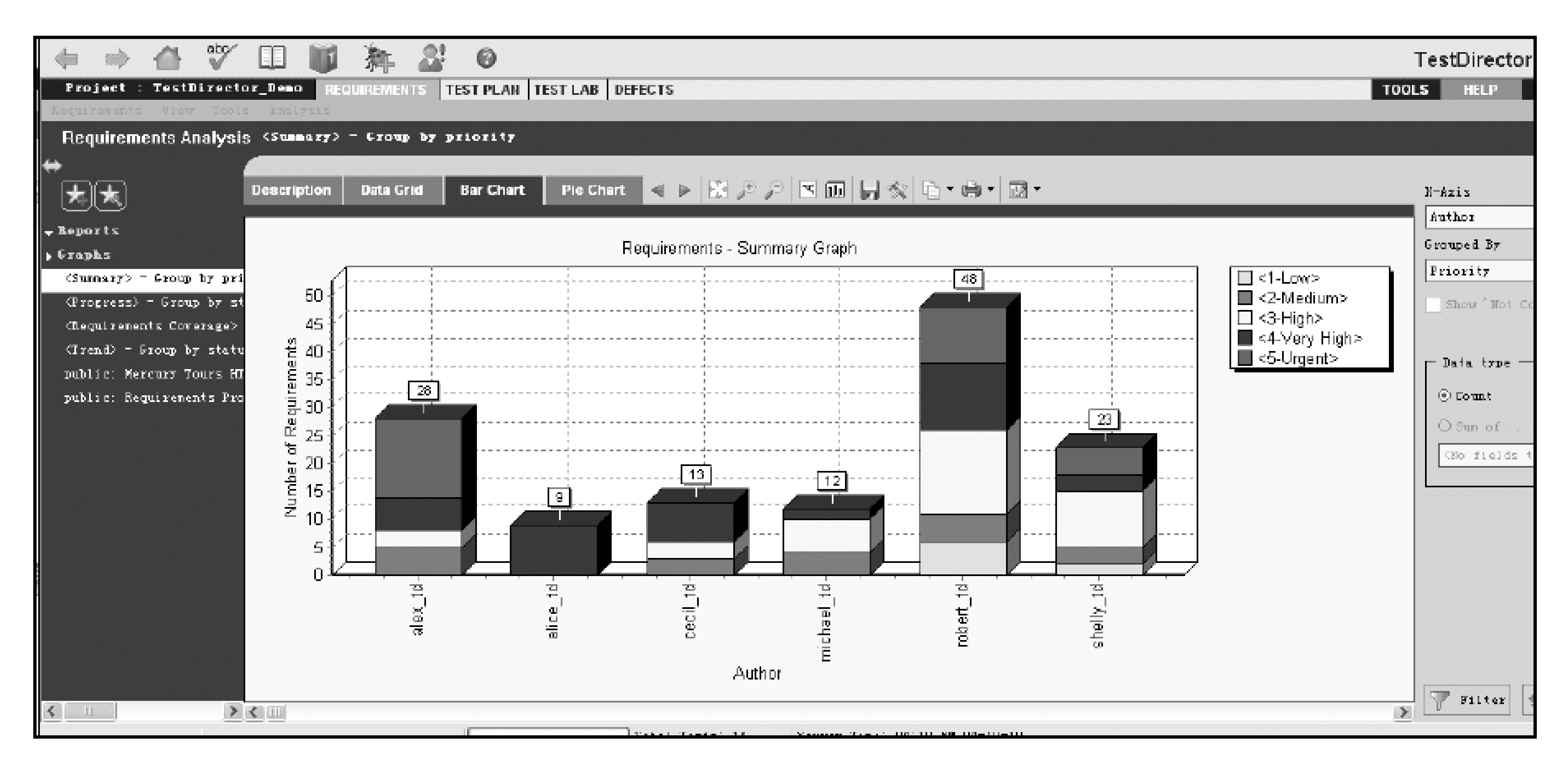


图 4.13 需求的优先级分布图

测试人员都知道，最终判断软件是否正确必须依据用户的需求。而 TestDirector 就是通过上面所说的测试需求管理模块来管理和维护这一测试的根本依据的。

3）管理测试用例

在分析和明确了测试需求后，需要进一步地详细设计覆盖需求的测试用例，用于指导测试人员的测试执行。测试用例的设计是对将来的测试活动进行计划的一个过程，TestDirector 通过 TEST PLAN 模块来管理测试用例。测试用例的管理可以通过树状结构的层次关系来组织，如图 4.14 所示。

下面是一个添加测试用例的操作过程：

（1）选择 Planning→New Folder→New Test 命令，来创建测试用例分类目录和测试用例。

（2）选中某个测试用例，可在右边的 Details 标签中编辑该测试用例的描述，例如测试用例的目的、测试用例输入的参数等。

（3）在 Design Steps 标签中可编辑测试用例的测试步骤、预期结果，如图 4.15 所示。

（4）在 Attachments 标签中可以加入一些必要的附件，例如测试参数文件或测试数据文件，还可以加入 URL 地址、屏幕截图、系统信息等内容。

（5）在 Reqs Coverage 标签中可以单击 Select Req 加入本测试用例覆盖的需求项，如

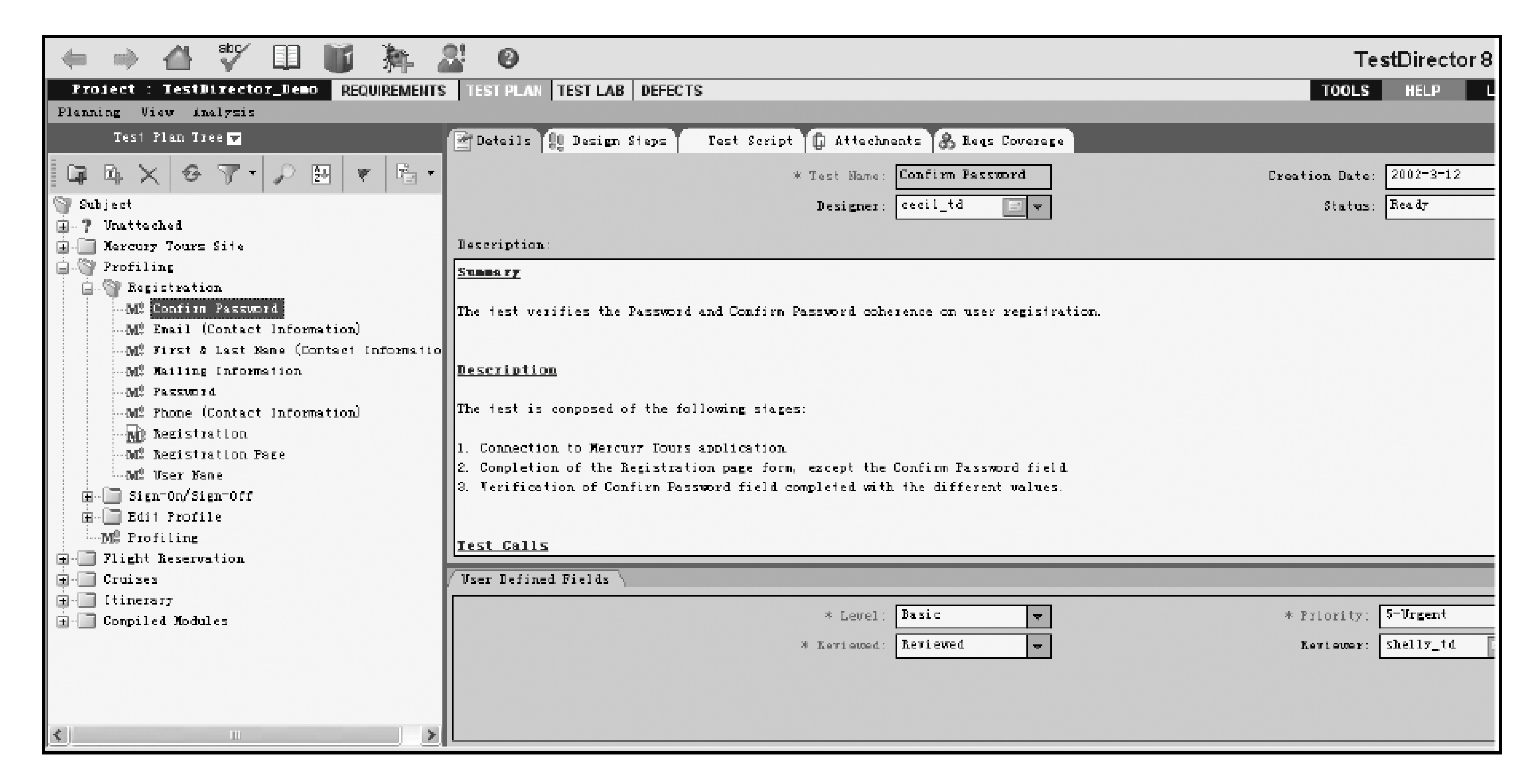


图 4.14 TestDirector 测试用例管理界面

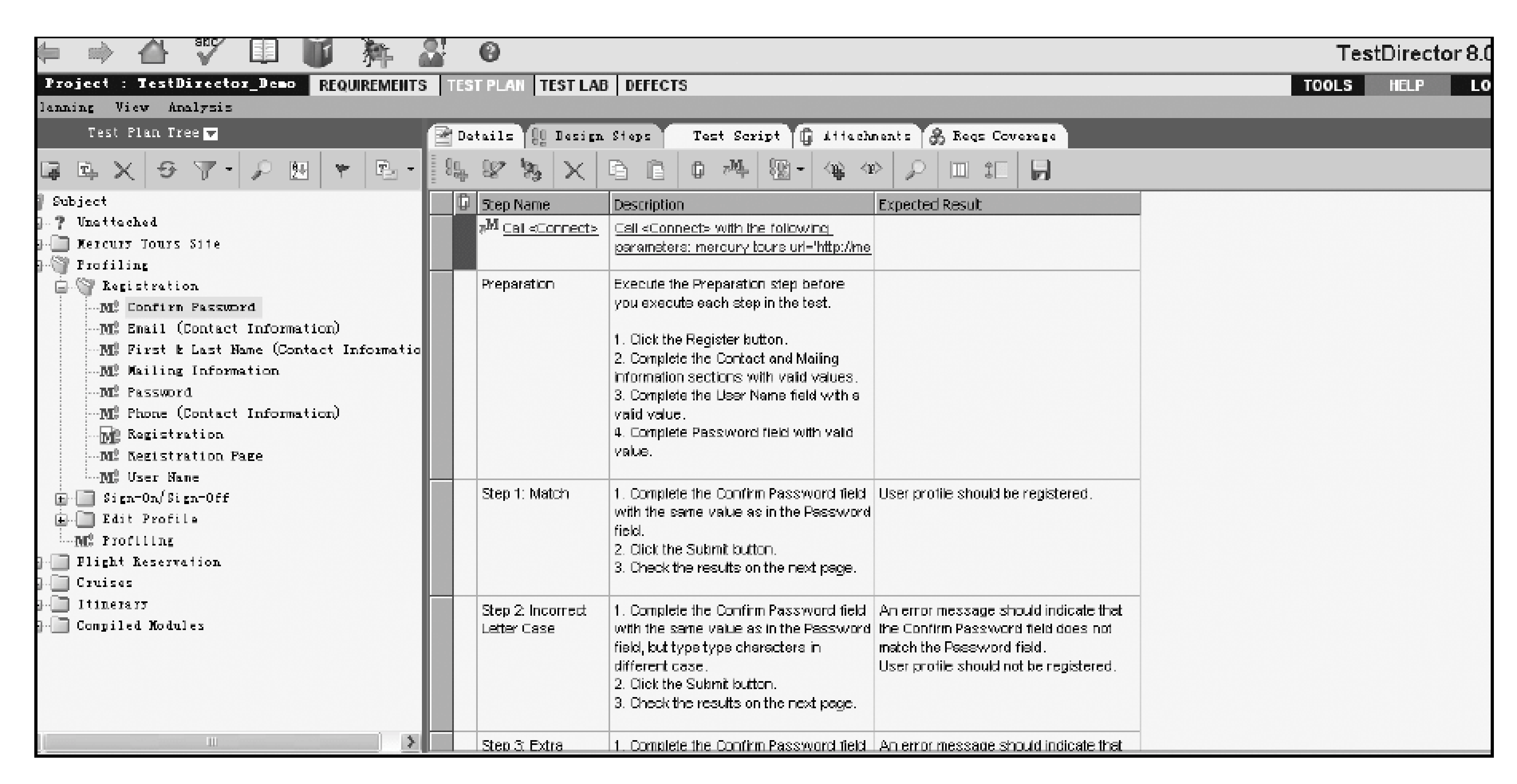


图 4.15 TestDirector 测试用例中的测试步骤

图 4.16 所示。

在测试用例设计模块，可通过选择 Analysis 命令来选择各种关于测试用例的报告。测试用例是测试的设计结果，TestDirector 通过测试计划模块的功能来管理和维护测试用例，形成测试项目的测试用例库。

4）管理测试过程

在设计好测试用例后，就可以在测试执行时选择需要参与测试的测试用例，打包成一个测试的集合，并按照集合中的每一项测试用例进行测试。这个过程的管理在 TestDirector 中是用 TEST LAB 模块来实现的，如图 4.17 所示。

下面是一个创建测试过程并执行测试的操作步骤：

（1）在界面左边的 Test Sets 编辑区域添加需要执行的测试集合。选择 Test Sets→New Test Set 命令，就可以添加一个空的测试集合。这是一个计划测试任务的过程，一

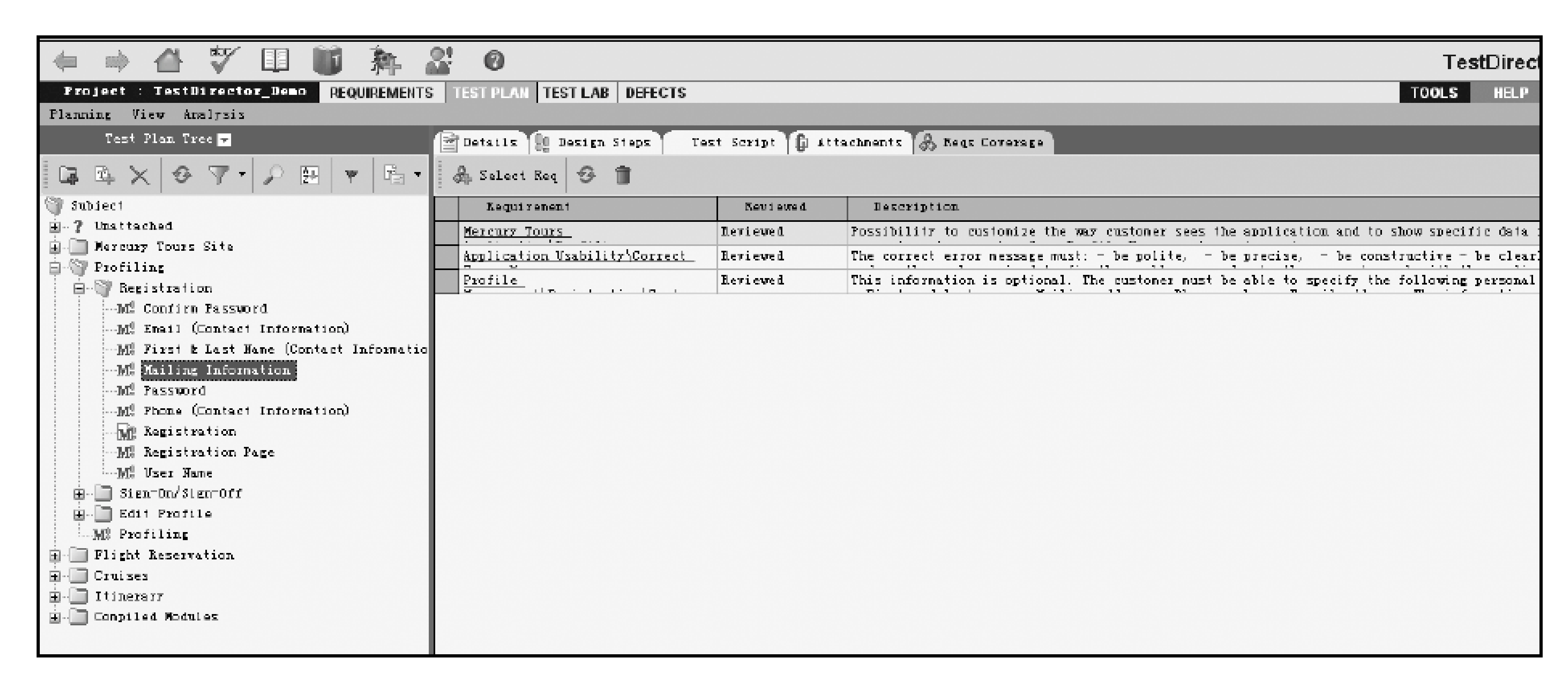


图 4.16 TestDirector 测试用例中的覆盖项

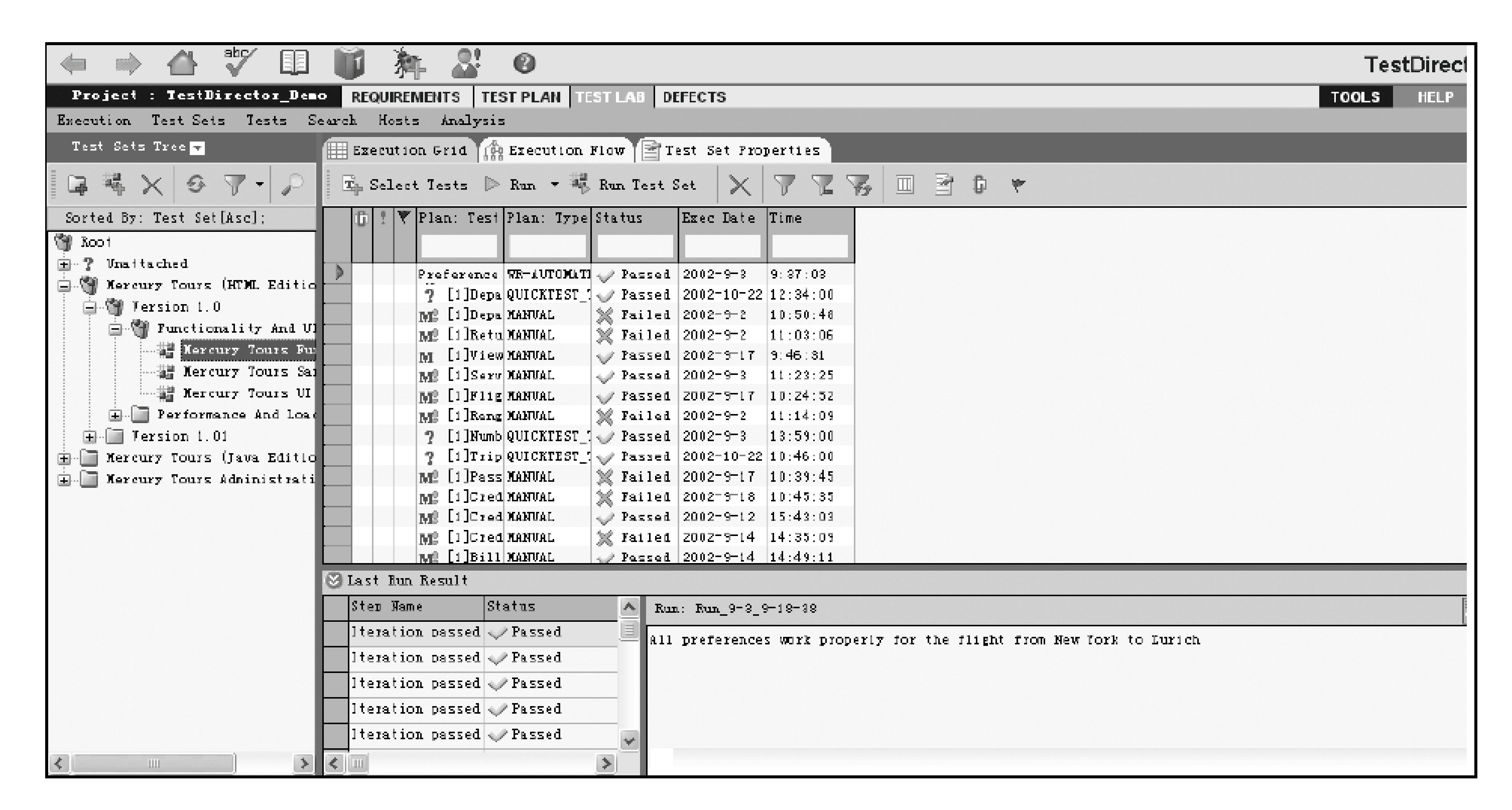


图 4.17 TestDirector 测试管理界面

般由测试组长负责编辑测试集合，并选择需要在这次测试任务中执行的测试用例。

（2）在右边区域的 Execution Grid 页，单击 Select Tests 按钮，可为测试集合添加需要执行的测试用例。

（3）添加了需要的测试用例后，测试组长在 Responsible Tester 列中指定测试用例由谁执行。

（4）指定计划执行的时间，如图 4.18 所示。

（5）负责按该测试用例执行测试的人在测试过程中，需要在 Status 列更新测试用例的执行状态，还需要指定真正执行的日期和时间。

（6）单击 Run 按钮，一步一步地执行测试，并记录测试结果。

在测试过程管理模块，可通过 Analysis 菜单选择各种关于测试过程和进度的报告。测试过程管理是考验测试人员尤其是测试管理者安排测试任务和分工的技巧以及选择测试用例的策略。测试过程也是测试工作的记录过程，是测试人员工作过程的体现。

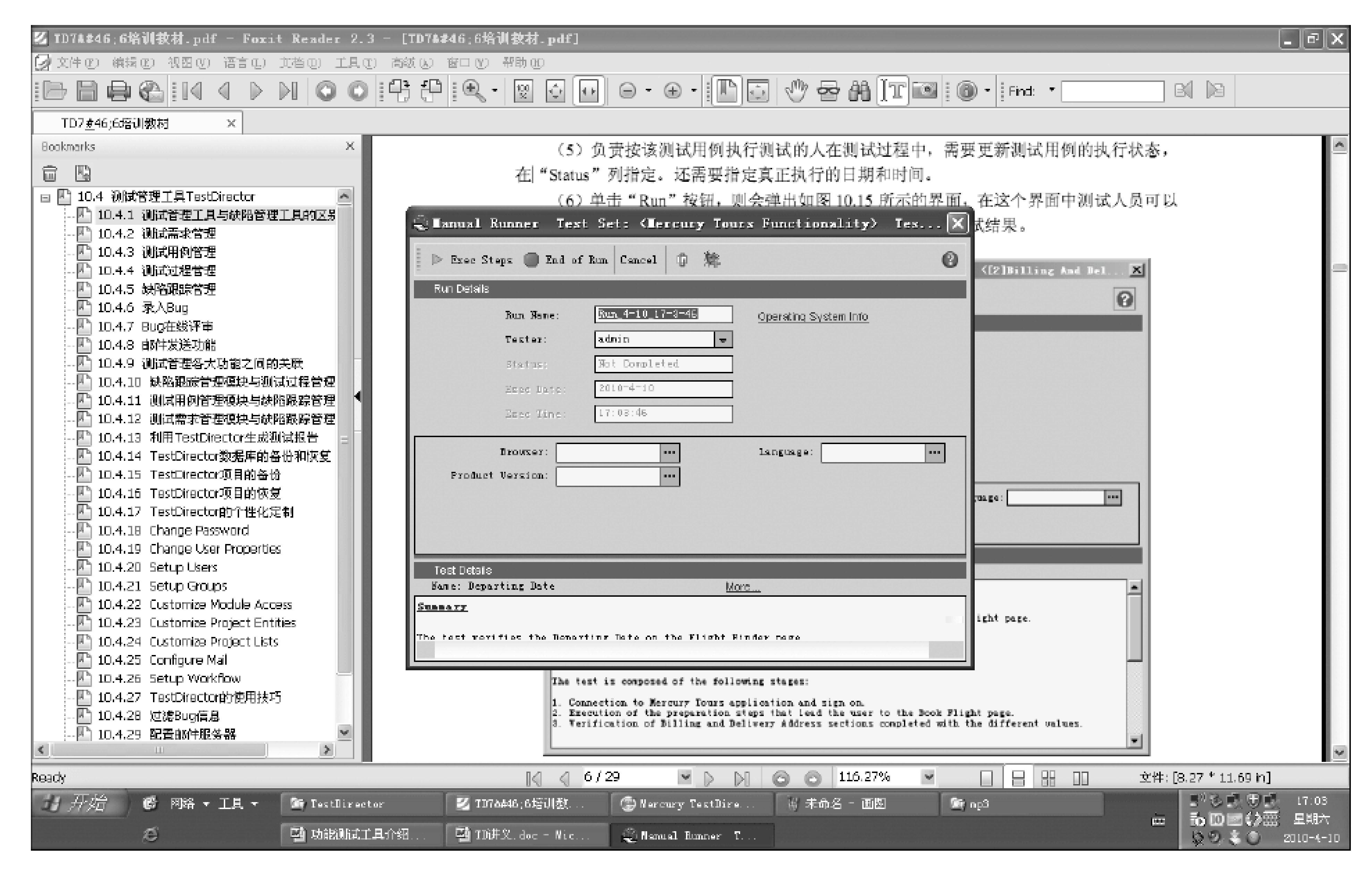


图 4.18 TestDirector 测试执行界面

5）管理测试缺陷

缺陷跟踪管理功能是测试管理工具的主角，也是测试人员日常工作最集中的地方，并且也是测试人员与开发人员沟通的基础平台。缺陷跟踪管理功能在 TestDirector 中是通过 DEFECTS 模块来实现的，如图 4.19 所示。

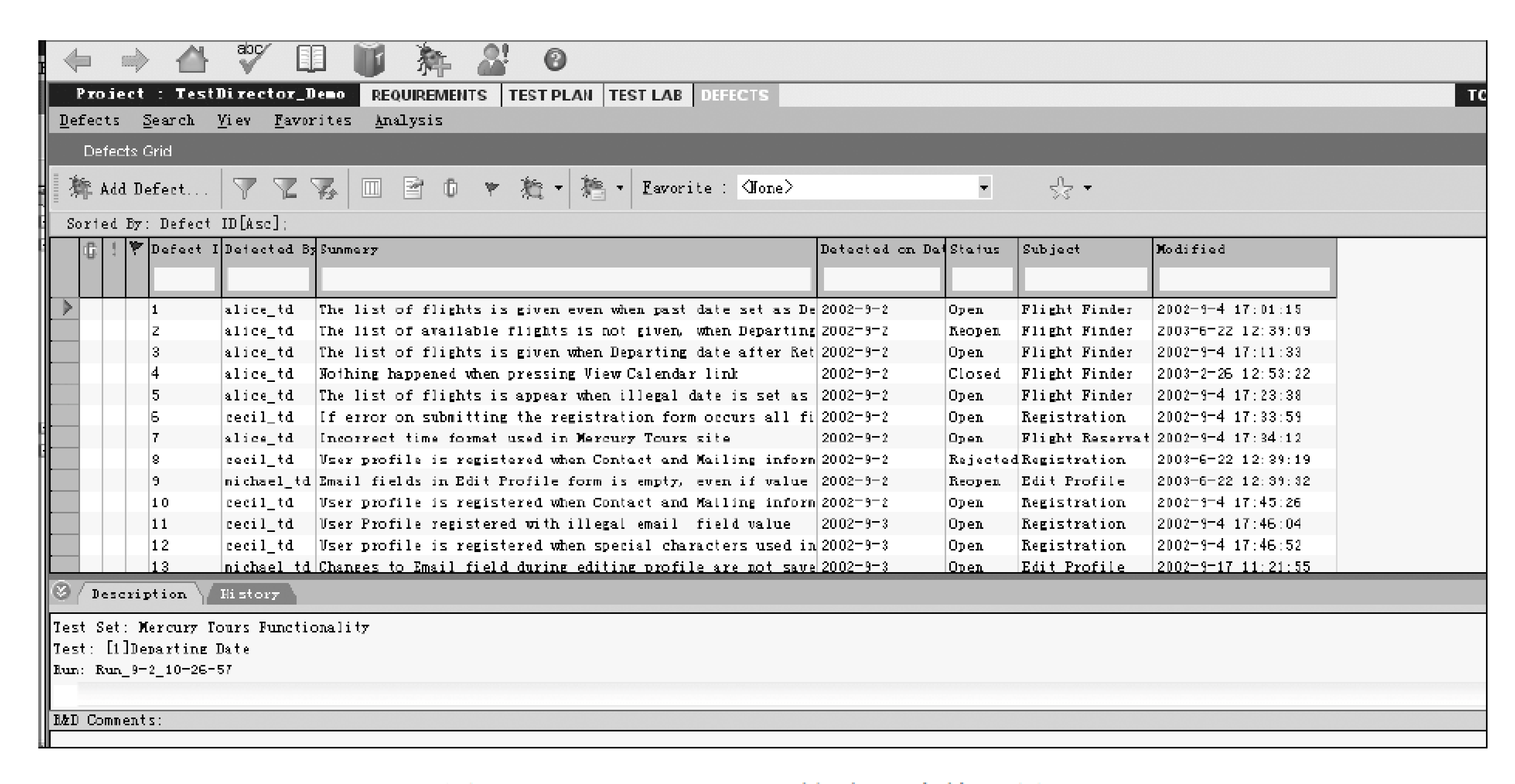


图 4.19 TestDirector 缺陷跟踪管理界面

6）录入缺陷

选择 Defects→Add Defect 命令，可以往缺陷列表中添加一条缺陷记录，如图 4.20 所示，在这个界面上，测试人员可以将发现的缺陷的相关信息录入到缺陷列表中。

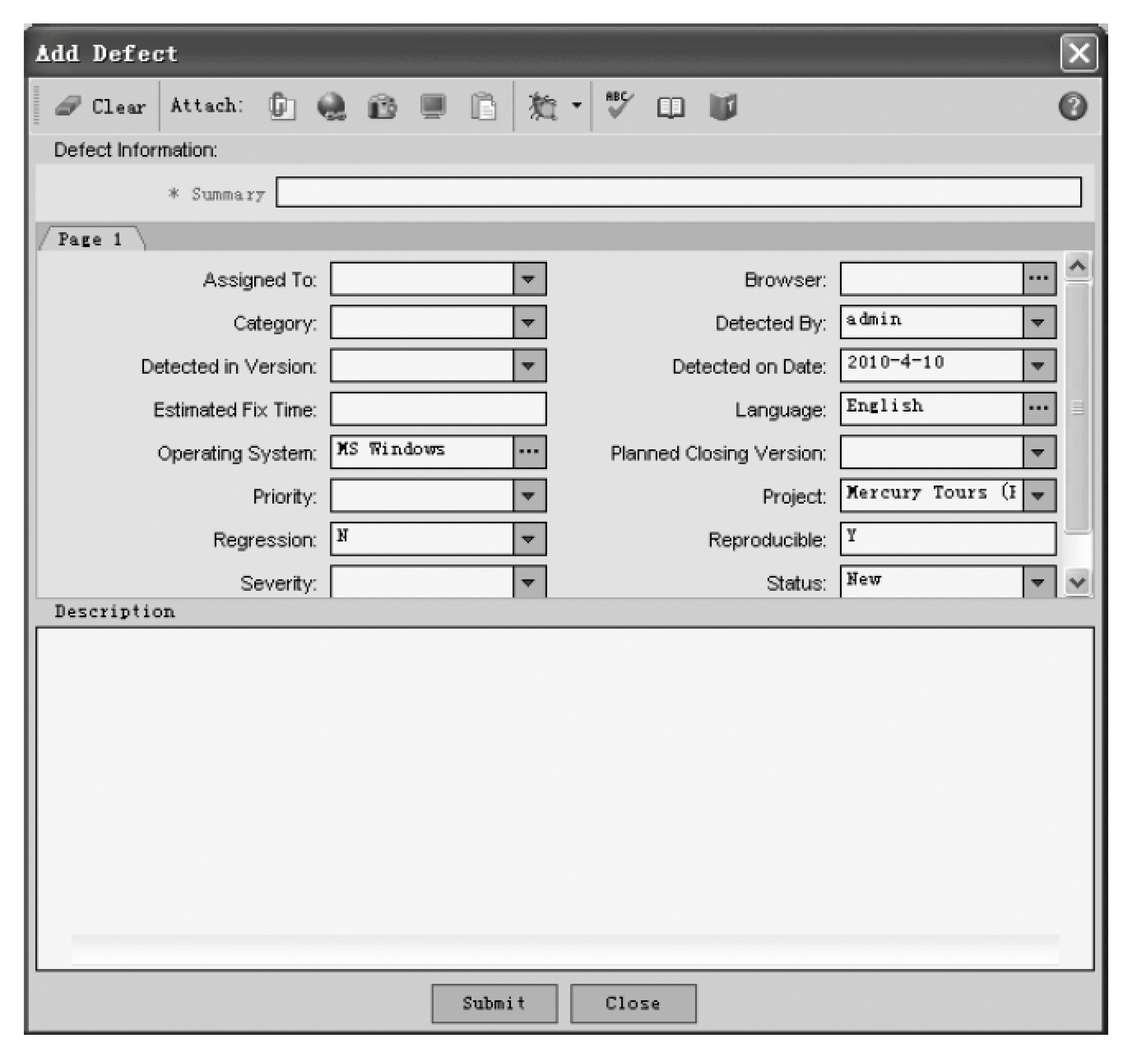


图 4.20　缺陷录入界面

一般需要录入的信息应该包括以下字段的内容：

- Summary，缺陷的简要描述。
- Assigned To，分派给谁负责处理这个缺陷。
- Detected By，这个缺陷是由谁发现的。
- Detected in Version，这个缺陷是在哪个版本发现的。
- Severity，缺陷的严重级别。
- Project，缺陷出现在哪个项目。
- Subject，缺陷出现在哪个功能模块。
- Status，一般新录入缺陷时把缺陷状态设置为 New。
- Description，缺陷的详细描述。

把缺陷正确录入后，接下来的工作就是跟踪缺陷的修改状态直到关闭。不同的项目组成员或角色可以使用不同的功能来对缺陷记录进行管理和维护。

7）利用 TestDirector 生成测试报告

在 TestDirector 的每个模块中都会有一个 Analysis 的菜单，用于查看和定制各种类型的分析报告。报告分成两大类，一类是 Reports 子菜单下的报告，另一类是 Graphs 子菜单下的报告，分别代表记录型和分析型两种测试报告。

（1）记录型报告是指把所有过程记录数据罗列出来，形成列表。在测试需求管理模块中，选择 Reports 子菜单下的 Standard Requirements Report 命令，则会出现图 4.21 所示的标准需求报告。

（2）分析型报告是指通过统计图表、趋势图、状态图等形式对数据进行分析和统计。

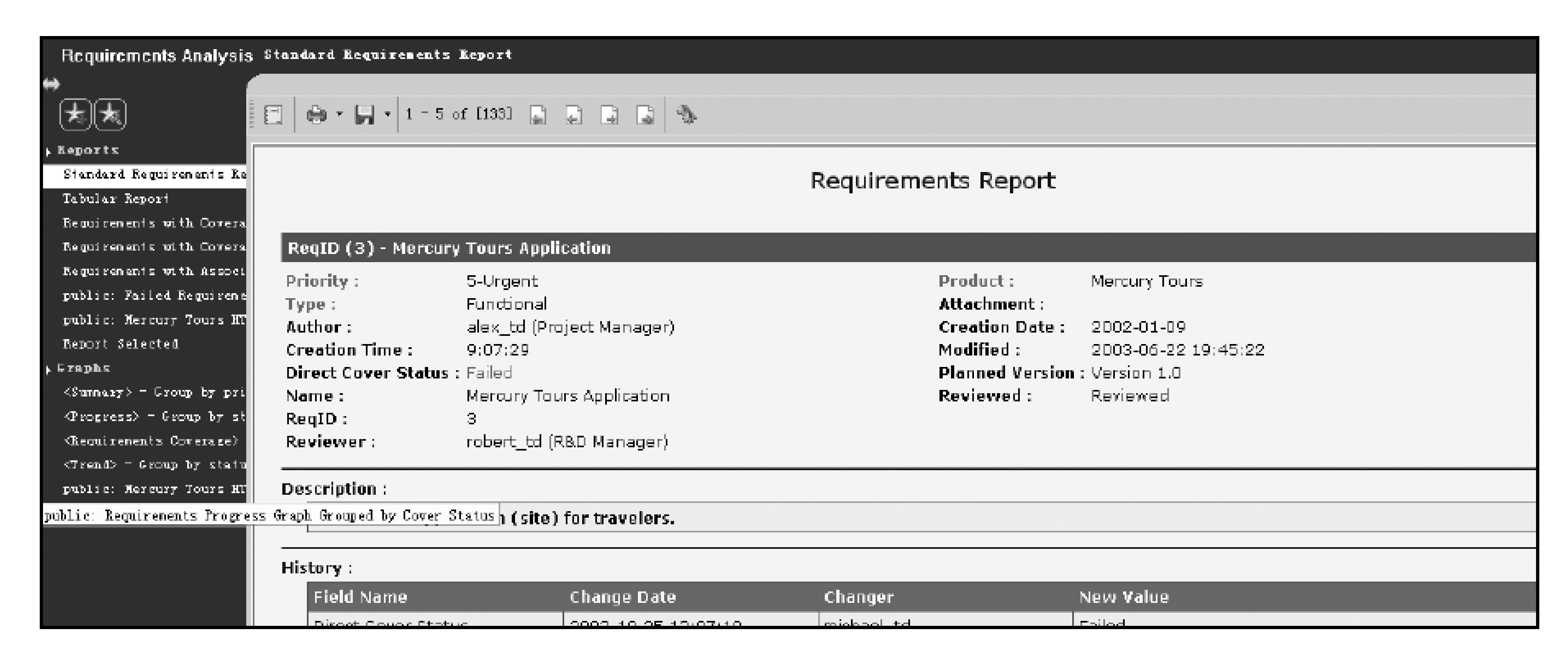


图 4.21 标准需求报告

这种报告的特点是能让别人看到整体的、全局的、高层的、形象的、直观的报告。例如，在测试用例管理模块，选择 Graphs 子菜单下的 Summary – Group by Status 命令，则出现如图 4.22 所示的报告。

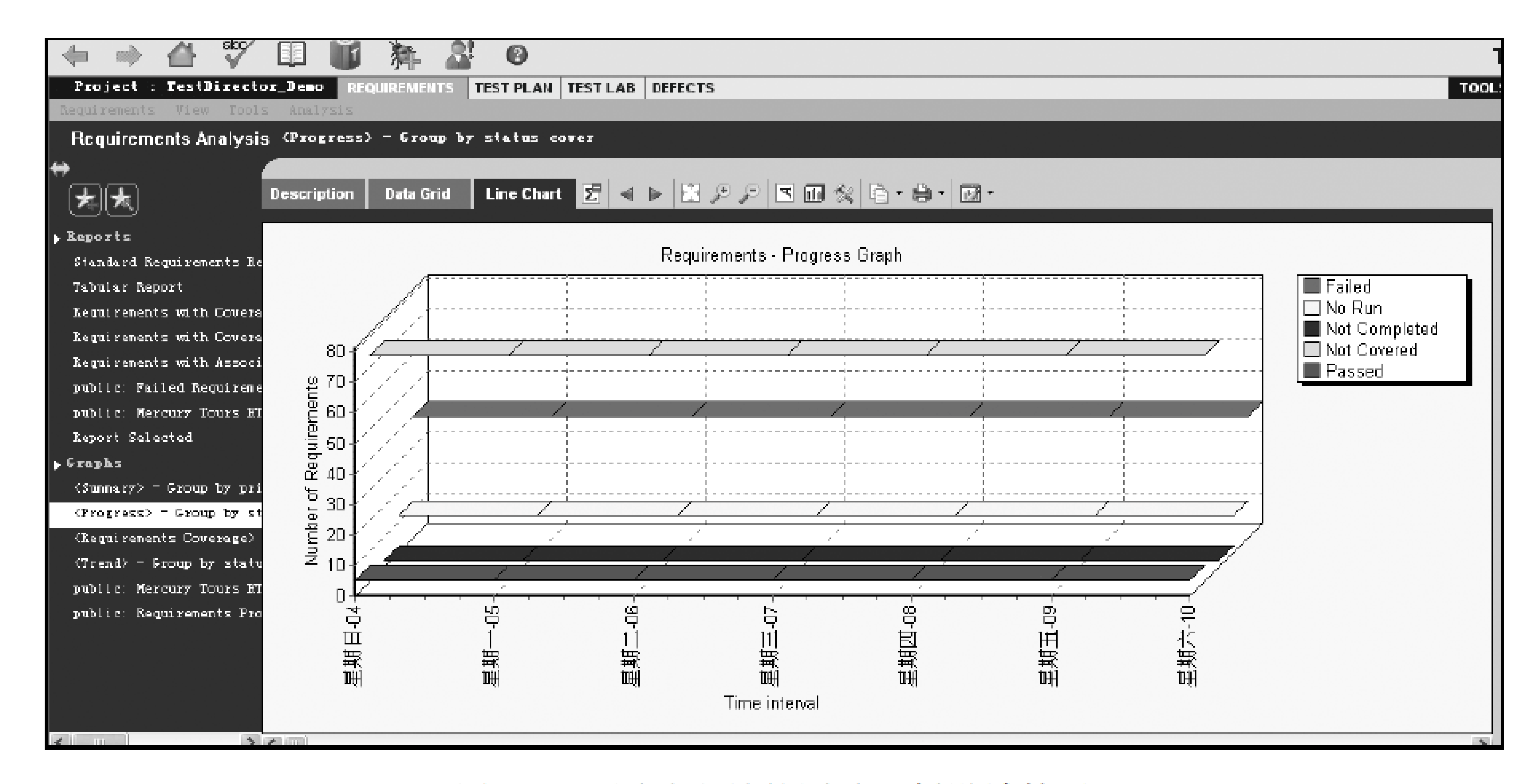


图 4.22 测试人员的测试用例设计情况

两种类型的报告各有特点，全面的测试报告应该包括这两种类型的报告。让不同的项目组成员可以找到自己关心的报告区域。例如，项目经理可能更关心高层的分析总结性的数据，而开发人员和测试人员可能需要仔细分析基础性的数据。



功能测试软件

实验目的：

(1) 理解功能测试软件的测试原理。

(2) 采用 VB 6.0 实现 GUI 捕捉/回放。

(3) 熟练掌握 UFT 的操作步骤。

实验环境： VB 6.0 软件、UFT 软件。

5.1 VB 6.0 实现 GUI 捕捉/回放

1. 实验原理

Windows 操作系统采用事件驱动的机制，通过消息的传递来实现功能。其中，钩子函数作为 Windows 系统的重要接口，用于监视系统或进程中的各种事件消息，比如截获键盘和鼠标的输入、屏幕取词、日志监视等，通过截获发往目标窗口的特定事件实现特定的功能。

本实验采用 VB 6.0 软件，通过调用钩子函数实现录制-回放功能。

2. 实验步骤

步骤 1：打开 VB 6.0 IDE，创建 Form 表单，其上有 3 个 Command 命令按钮、1 个 Label 和 1 个 Timer 控件，设置 Label 的 caption 属性为 00:00:00，Timer 的 Interval 属性为 500，如图 5.1 所示。

步骤 2：编写模块文件和窗体文件代码。

模块文件内容：

```
Option Explicit
Public Type EVENTMSG
        message As Long
        paramL As Long
        paramH As Long
        time As Long
        hwnd As Long
End Type
```

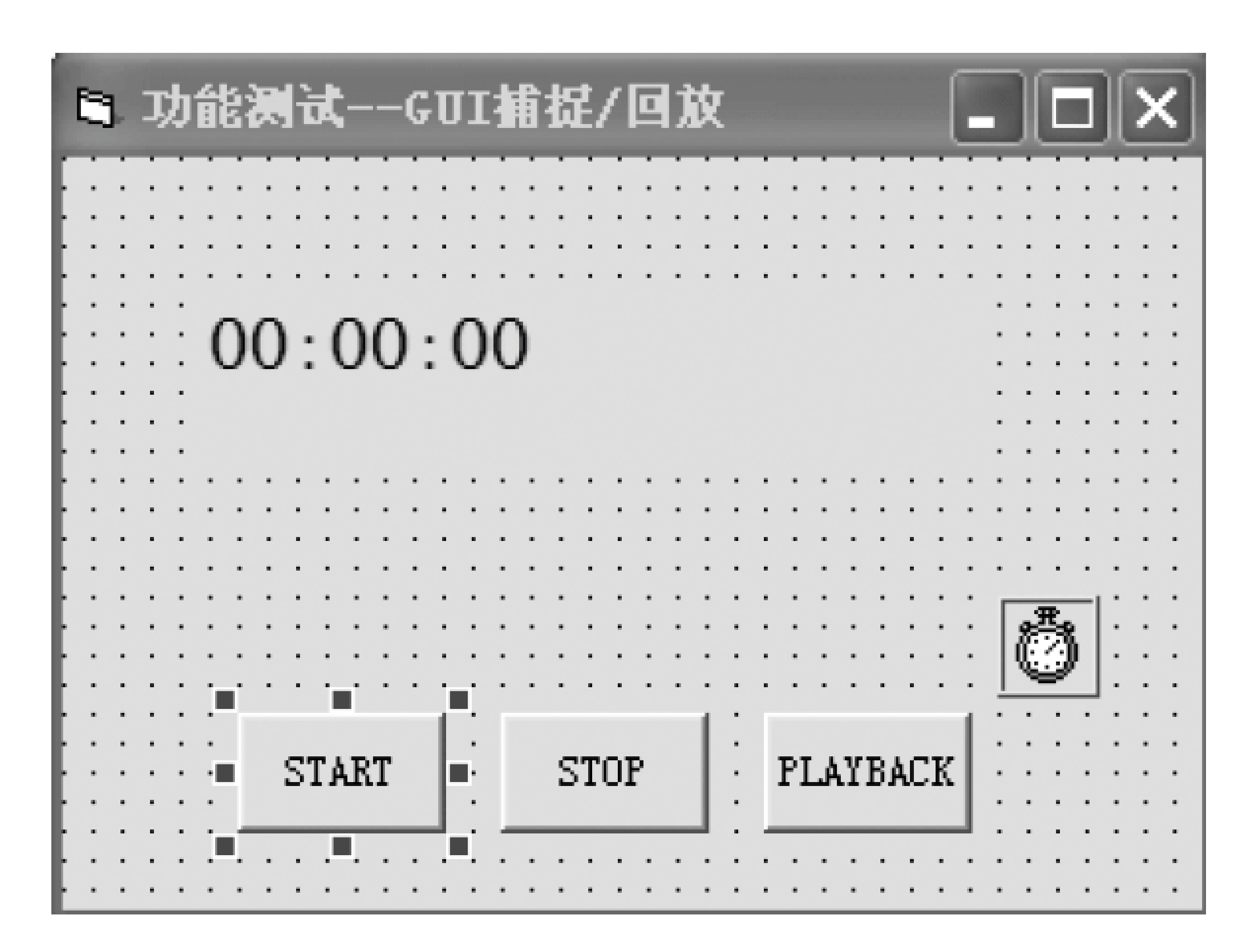


图 5.1 VB 6.0 设计界面

```
Public Declare Function CallNextHookEx Lib "user32"(ByVal hHook As Long, ByVal
    ncode As Long, ByVal wParam As Long, lParam As Any)As Long
Public Declare Function SetWindowsHookEx Lib "user32" Alias "SetWindowsHookExA"
    (ByVal idHook As Long, ByVal lpfn As Long, ByVal hmod As Long, ByVal dwThreadId
    As Long)As Long
Public Declare Sub CopyMemoryT2H Lib "kernel32" Alias "RtlMoveMemory"(ByVal Dest
    As Long, Source As EVENTMSG, ByVal Length As Long)
Public Declare Sub CopyMemoryH2T Lib "kernel32" Alias "RtlMoveMemory"(Dest As
    EVENTMSG, ByVal Source As Long, ByVal Length As Long)
Public Declare Function UnhookWindowsHookEx Lib "user32"(ByVal hHook As Long)
    As Long
Public Const WH_JOURNALPLAYBACK=1
Public Const WH_JOURNALRECORD=0
Public Const HC_SYSMODALOFF=5
Public Const HC_SYSMODALON=4
Public Const HC_SKIP=2
Public Const HC_GETNEXT=1
Public Const HC_ACTION=0
Public EventArr(1000)As EVENTMSG
Public EventLog As Long
Public PlayLog As Long
Public hHook As Long
Public hPlay As Long
Public recOK As Long
Public canPlay As Long
Public bDelay As Boolean
Public Function HookProc(ByVal iCode As Long, ByVal wParam As Long, ByVal lParam
    As Long)As Long
    Dim Result As Long
```

```
    recOK=1
    Result=0
    If iCode<0 Then                         'iCode 小于 0,必须直接调用下一个消息钩子函数
        Result=CallNextHookEx(hHook, iCode, wParam, lParam)
    ElseIf iCode=HC_SYSMODALON Then         '不允许记录
        recOK=0
    ElseIf iCode=HC_SYSMODALOFF Then        '允许记录
        recOK=1
    ElseIf((recOK>0)And(iCode=HC_ACTION))Then        '将消息记录在记录队列中
        CopyMemoryH2T EventArr(EventLog), lParam, Len(EventArr(EventLog))
        EventLog=EventLog+1
        If EventLog >=1000 Then             '当记录大于 1000 后释放消息钩子
            UnhookWindowsHookEx hHook
        End If
    End If
    HookProc=Result
End Function
Public Function PlaybackProc(ByVal iCode As Long, ByVal wParam As Long, ByVal
    lParam As Long)As Long
    Dim Result As Long
    canPlay=1
    Result=0
   If iCode<0 Then                          'iCode 小于 0,必须直接调用下一个消息钩子函数
        Result=CallNextHookEx(hPlay, iCode, wParam, lParam)
    ElseIf iCode=HC_SYSMODALON Then         '不允许回放
        canPlay=0
    ElseIf iCode=HC_SYSMODALOFF Then        '允许回放
        canPlay=1
    ElseIf((canPlay=1)And(iCode=HC_GETNEXT))Then
        If bDelay Then
            bDelay=False
            Result=50
        End If
        '从记录队列中取出消息并赋予 lParam 指针指向的 EVENTMSG 区域
        CopyMemoryT2H lParam, EventArr(PlayLog), Len(EventArr(EventLog))
    ElseIf((canPlay=1)And(iCode=HC_SKIP))Then
        bDelay=True
        PlayLog=PlayLog+1
    End If
    If PlayLog >=EventLog Then
        UnhookWindowsHookEx hPlay
    End If
    PlaybackProc=Result
End Function
```

Form 窗体文件如下：

```
Public dtStart As Date
Private Sub Command1_Click()
EventLog=0
    hHook=SetWindowsHookEx(WH_JOURNALRECORD, AddressOf HookProc, _App.hInstance, 0)
    Command2.Enabled=True
    Command1.Enabled=False
    Timer1.Enabled=True
    dtStart=time()
End Sub
Private Sub Command2_Click()
        UnhookWindowsHookEx hHook
    hHook=0
    Command1.Enabled=True
    Command2.Enabled=False
    Command3.Enabled=True
    Timer1.Enabled=False
End Sub
Private Sub Command3_Click()
PlayLog=0
    hPlay=SetWindowsHookEx(WH_JOURNALPLAYBACK, AddressOf PlaybackProc, _App.hInstance,
         0)
    Command3.Enabled=False
End Sub
Private Sub Form_Load()
Command1.Caption="记录"
    Command2.Caption="停止"
    Command3.Caption="回放"
    Command2.Enabled=False
    Command3.Enabled=False
End Sub
Private Sub Timer1_Timer()
    lblTime.Caption=CDate(time()-dtStart)
End Sub
```

步骤 3：运行 VB 工程(F5 键)。单击 START 按钮，等待数秒，观察 Label 的数字变化。然后单击 STOP 按钮。最后单击 PLAYBACK 按钮，观察程序执行情况。

5.2　UFT

UFT(统一功能测试)原来称为 Quicktest Professional(QTP)，作为功能的回归自动化测试工具，针对 GUI 应用程序包括传统的 Windows 应用程序以及 Web 应用，不仅适用于开发早期，而且在有大量重复性的手工测试的项目、测试时间比较长的项目、回归测

试等流程中具有绝对的优势。

5.2.1 基本功能

UFT 自动化测试的基本功能如图 5.2 所示。

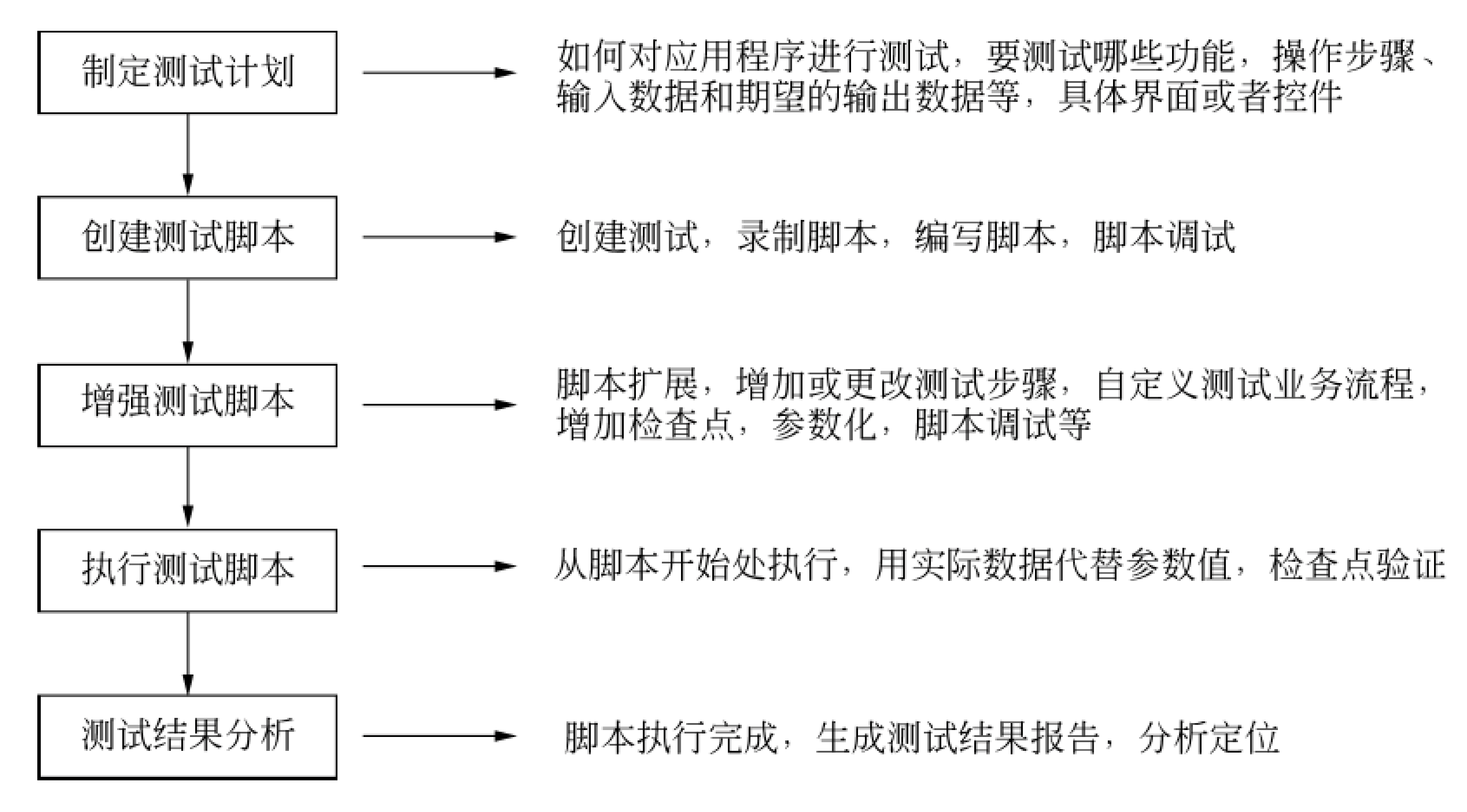


图 5.2　UFT 基本功能

5.2.2 安装 UFT

步骤 1：下载 UFT 安装包，如图 5.3 所示，下载地址：http://saas.hpe.com/en-us/software/uft，下载前需要进行账户注册，UFT 版本为 14.0。

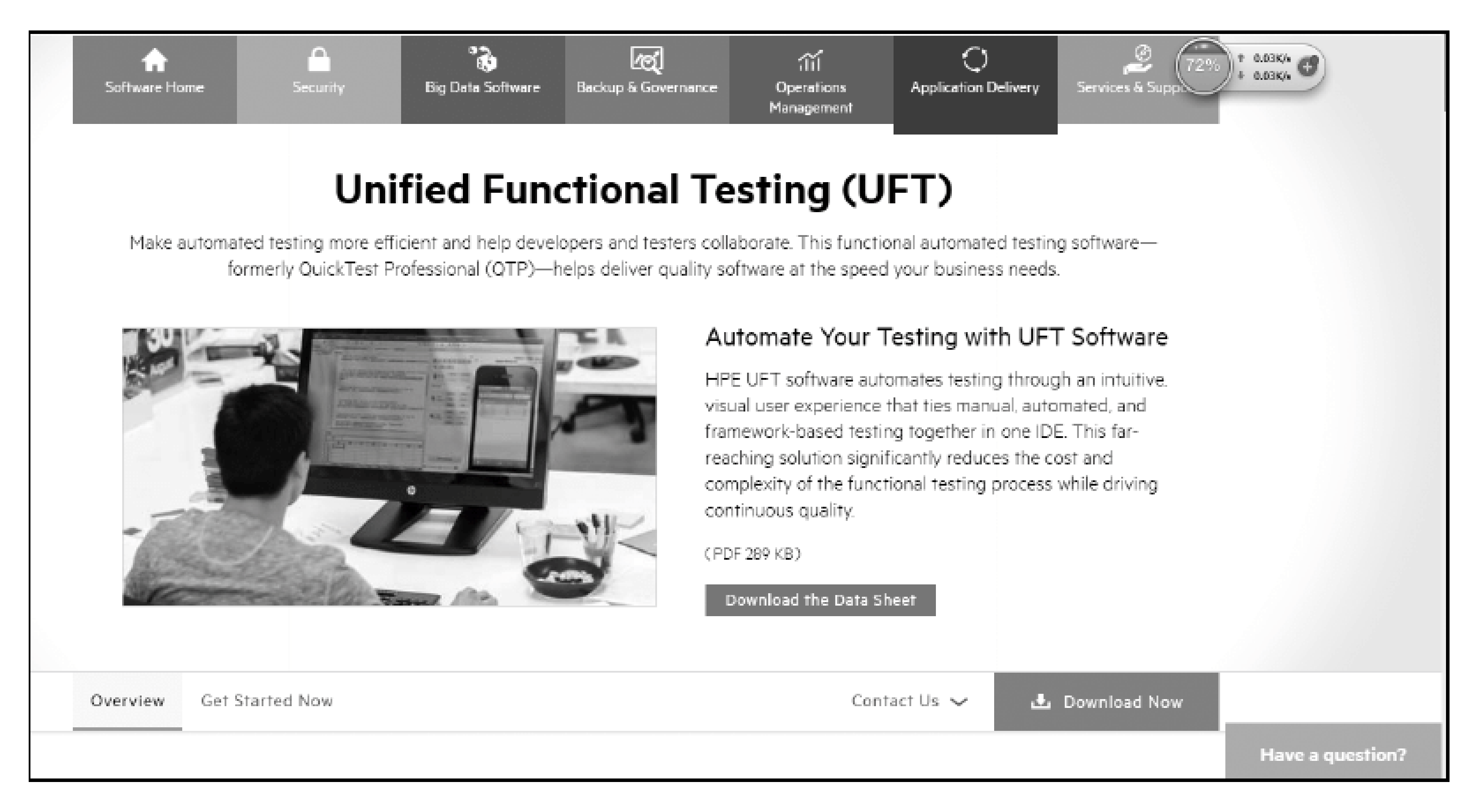


图 5.3　官方下载网页

步骤 2：执行 setup 程序开始安装，如图 5.4 所示。

步骤 3：打开安装页面，按照提示安装需要的功能，如图 5.5 所示。

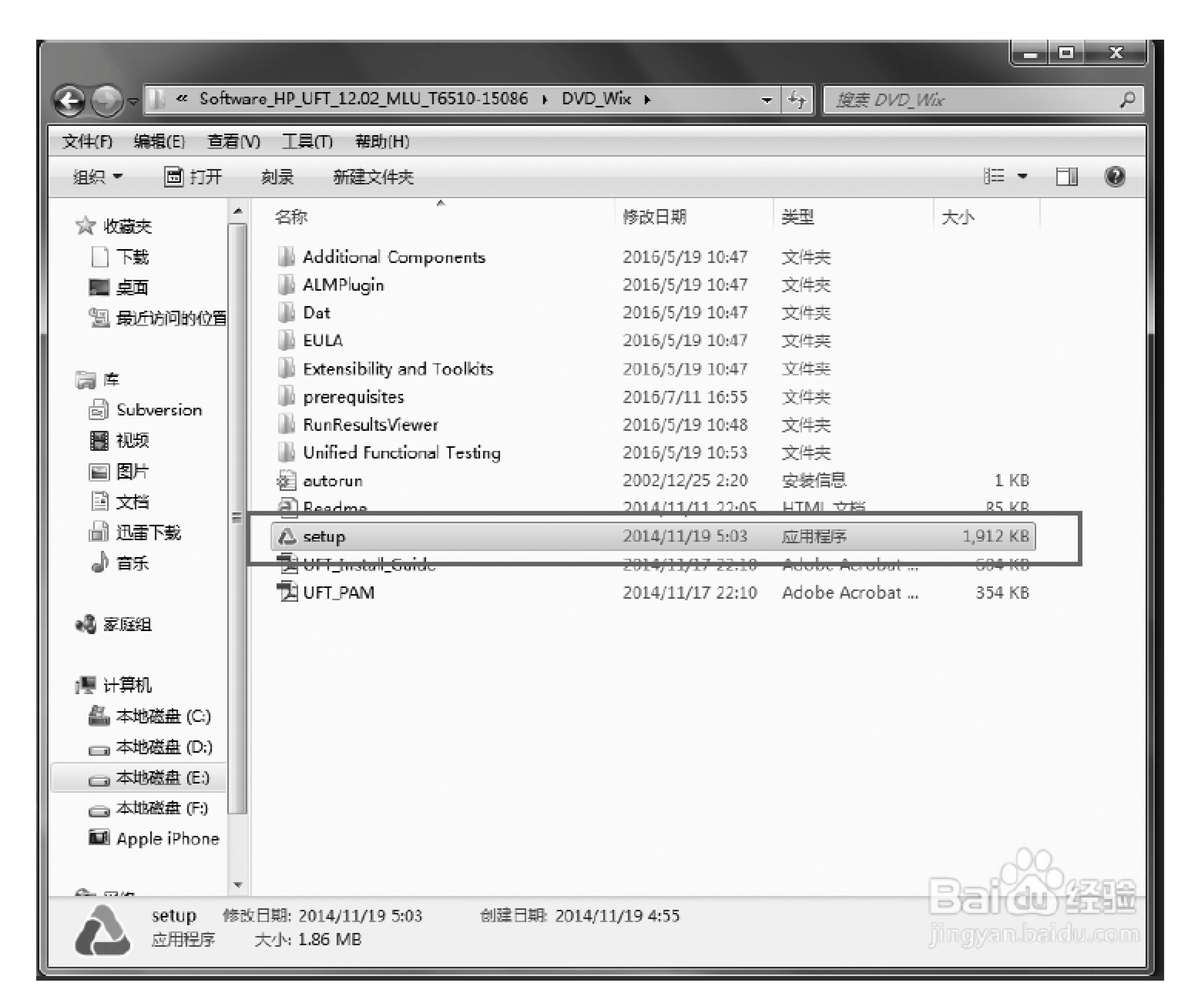


图 5.4　UFT 安装文件



图 5.5　UFT 安装页面

5.2.3　实验内容

1. UFT 基本功能操作

步骤 1：单击图标进入应用，弹出图 5.6 所示的许可证警告，单击“继续”按钮。

步骤 2：设置插件，如图 5.7 所示。

步骤 3：打开 UFT，显示图 5.8 所示的界面。

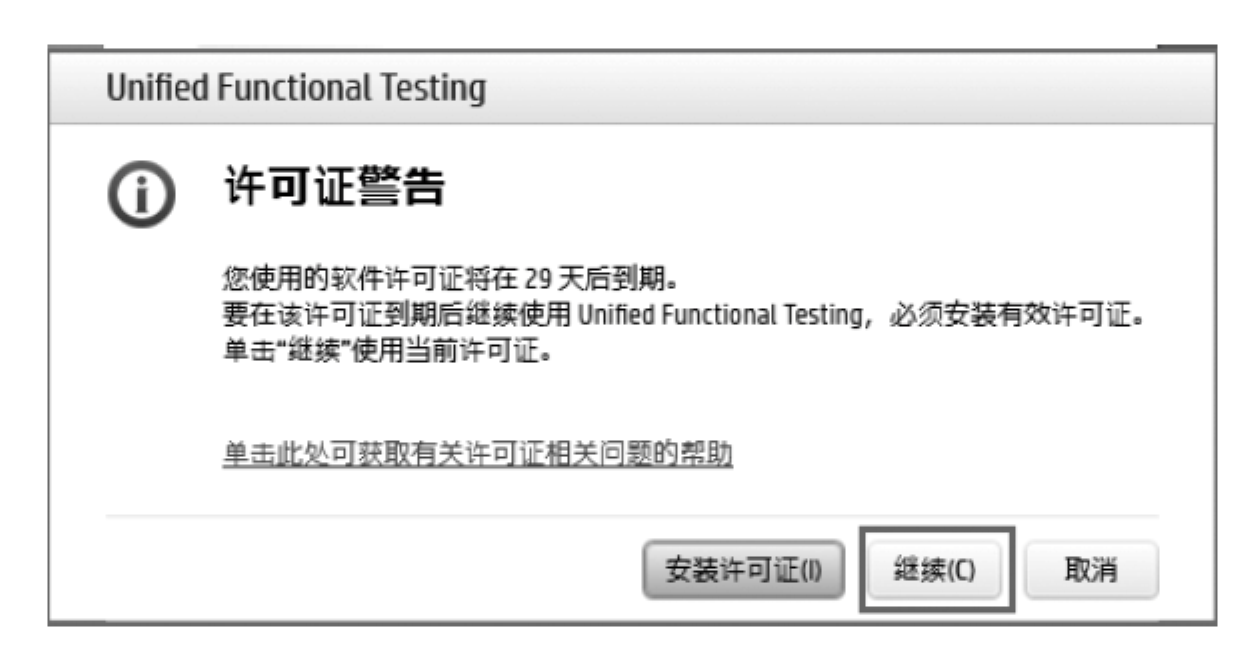


图 5.6　打开 UFT



图 5.7　设置插件



图 5.8　UFT 界面

步骤 4：单击"新建"按钮，新建 GUI 测试，如图 5.9 所示。



图 5.9　新建 GUI 测试

步骤 5：单击“创建”按钮，如图 5.10 所示。

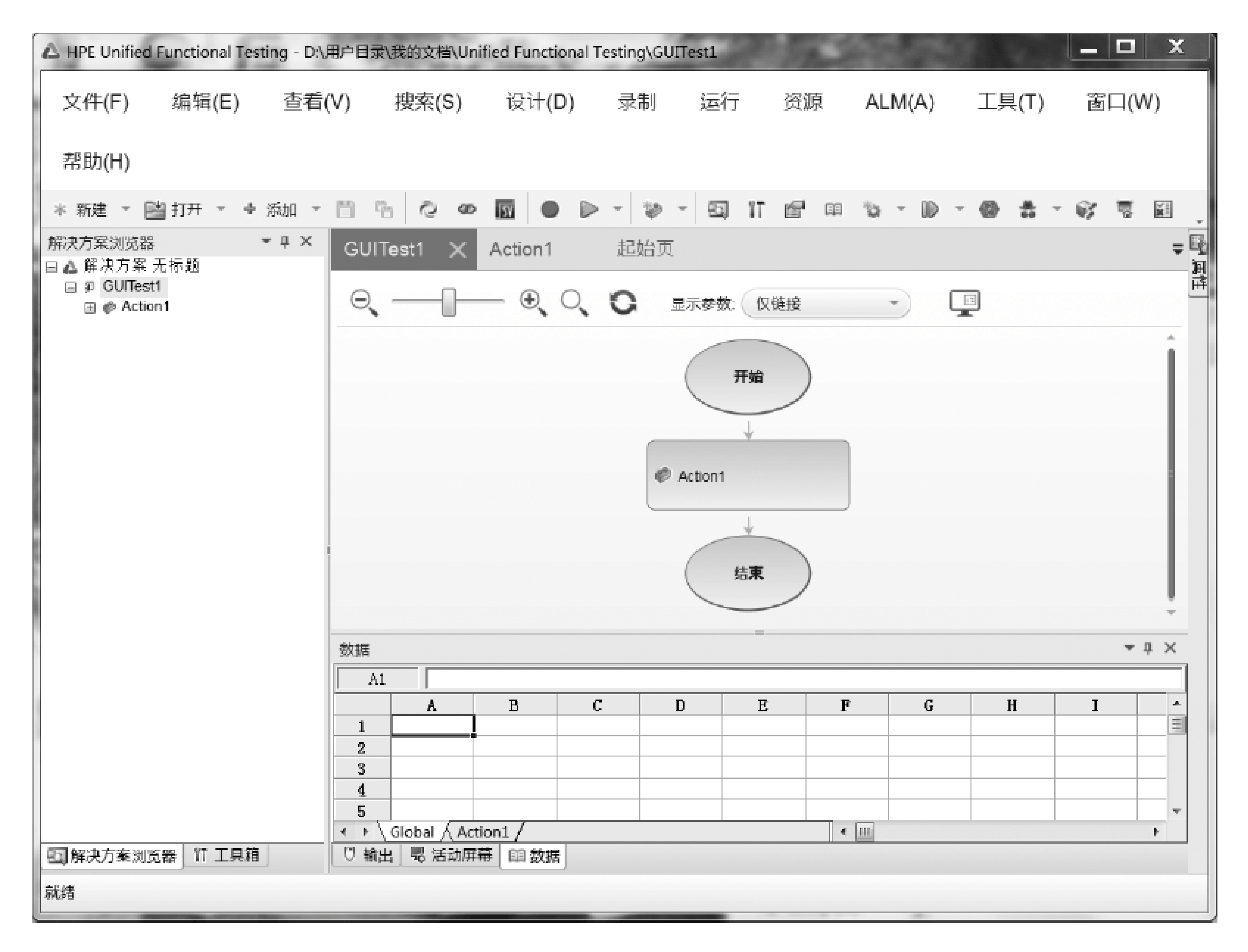


图 5.10　GUI 测试界面

步骤 6：选择“工具”菜单中的“选项”命令，如图 5.11 所示。

步骤 7：单击“GUI 测试”，在左侧单击“测试运行”，在普通模式下将每步执行延迟的秒数改为 1500，其他选项保持不变，如图 5.12 所示。

步骤 8：选择“录制”菜单中的“录制和运行设置”命令，如图 5.13 所示。

步骤 9：选择 Web 选项卡，选中“录制或运行时打开以下浏览器”，在 URL 栏中输入目标测试网址或 IP 地址。由于录制的网站带有验证码，而验证码在网站每次打开的都不相同，故录制运行后，将不能再现过程。因此，本次实验以“个人博客”为例，不需要验证

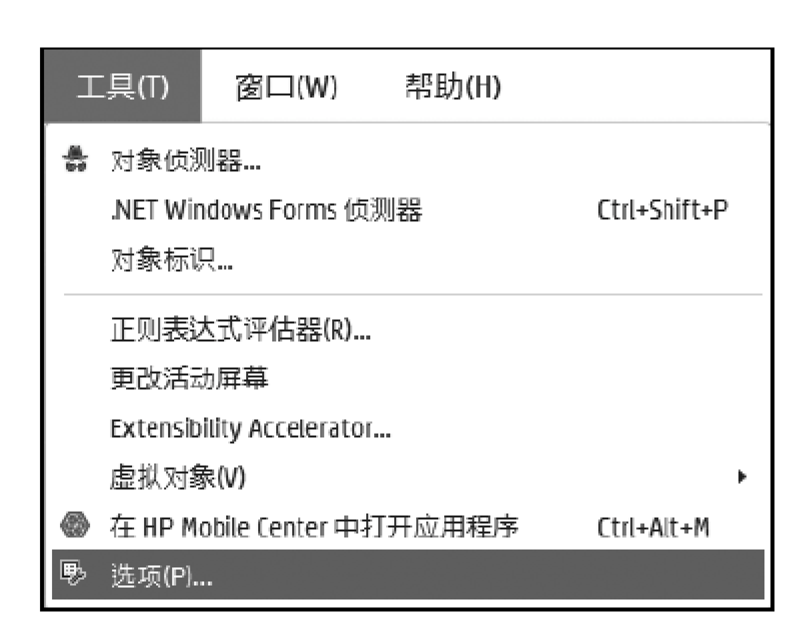


图 5.11　选择“工具”菜单“选项”命令



图 5.12　设置延迟秒数

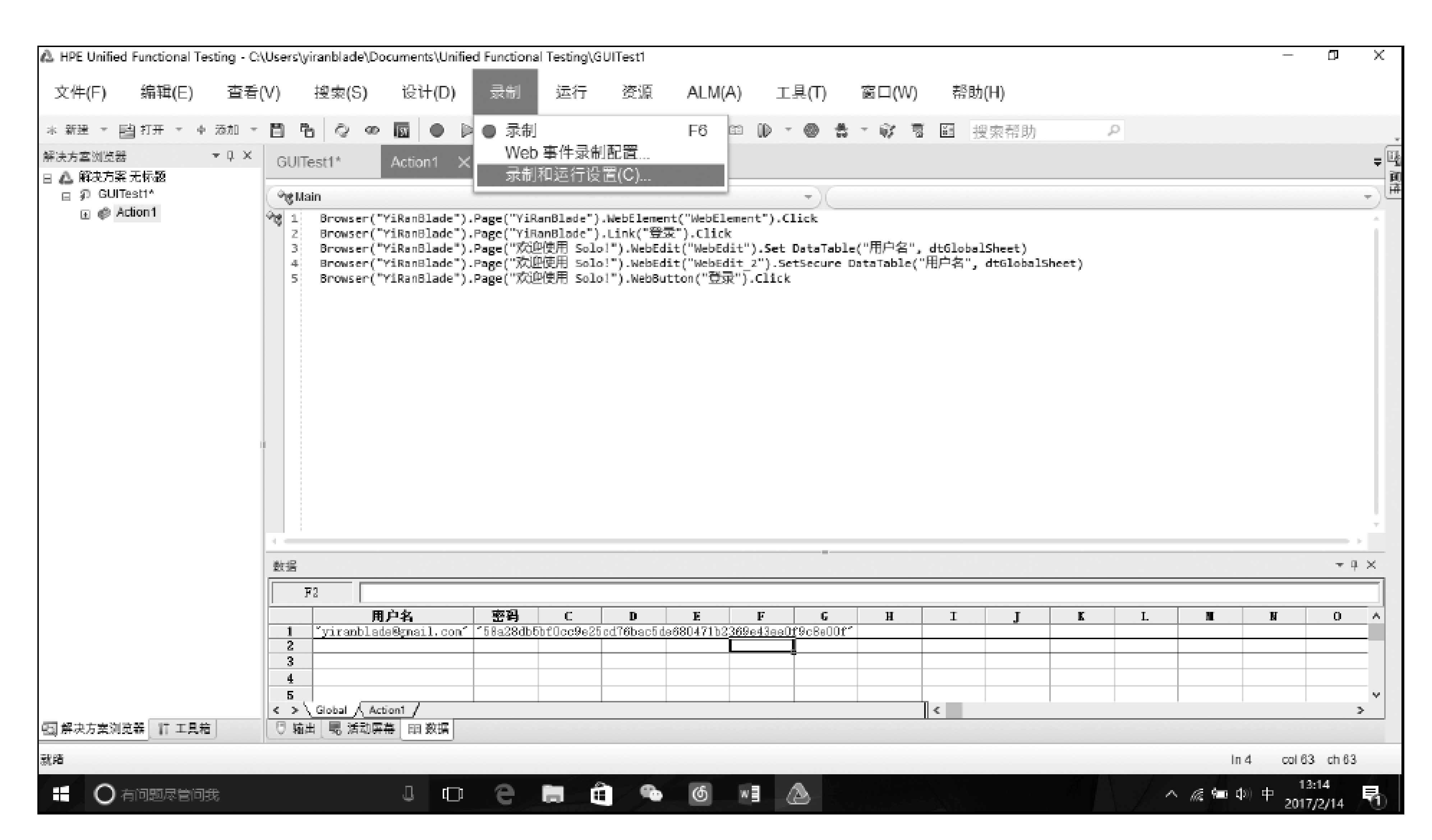


图 5.13　录制设置

码,输入网址 http://yiranblade.cn/login? goto=http%3A%2F%2Fyiranblade.cn%2Fadmin-index.do%23main,在"浏览器"栏中选择 Google Chrome,如图 5.14 所示。



图 5.14 录制和运行设置

步骤 10:单击"确定"按钮后,单击录制按钮,会自动打开 Chrome 浏览器并进入"个人博客"(此时录制的方式为"默认"方式),如图 5.15 所示。



图 5.15 UFT 录制

步骤 11:关闭页面后,单击停止录制按钮,如图 5.16 所示。



图 5.16 停止录制

步骤 12：选中 Action1，在此界面中显示本次录制的所有操作的脚本，可以通过修改和编写相关的代码来实现各种操作，如图 5.17 所示。

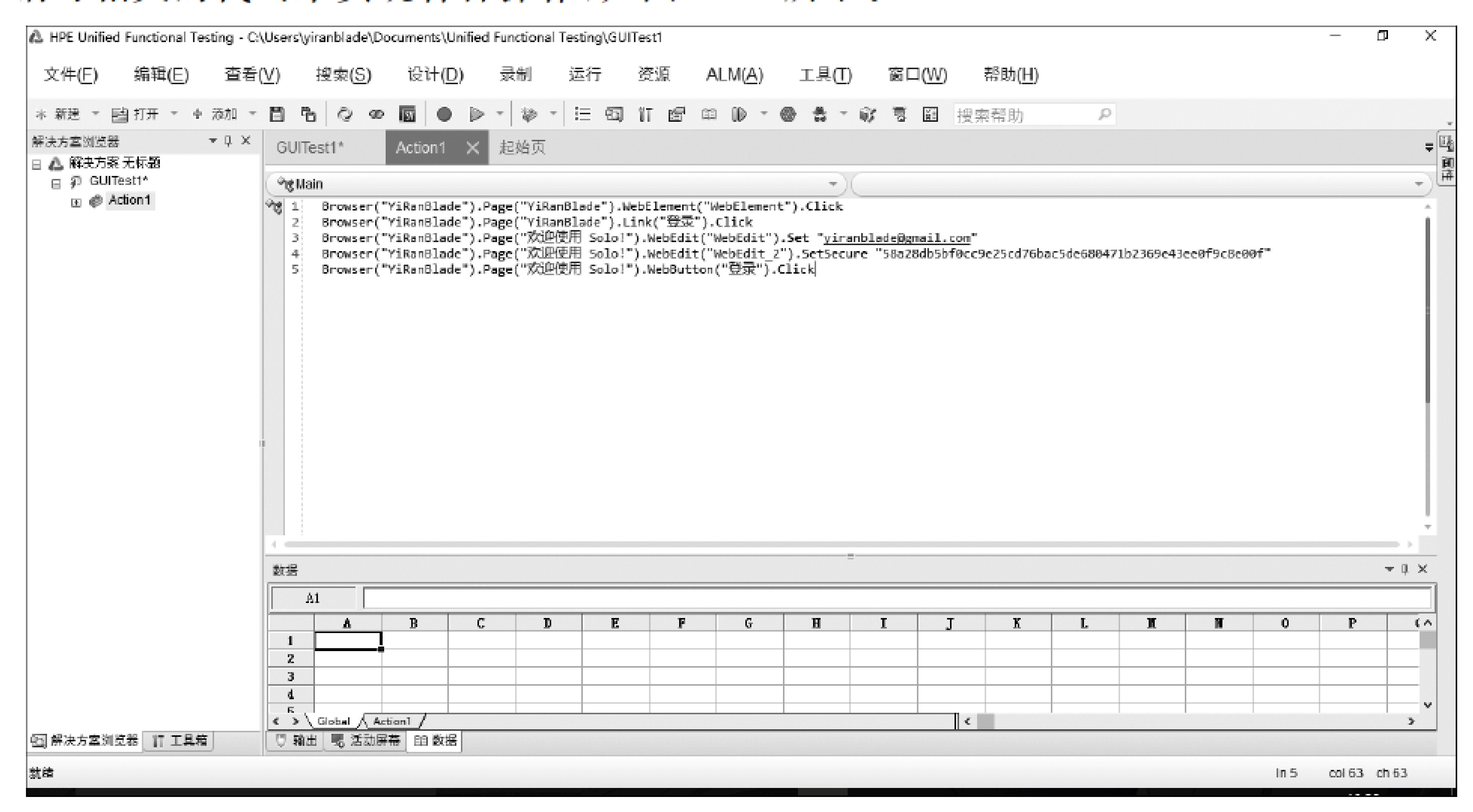


图 5.17　查看脚本

步骤 13：单击▷(运行)按钮或按快捷键 F5 运行录制的脚本，检验脚本是否能够运行成功，如图 5.18 所示。

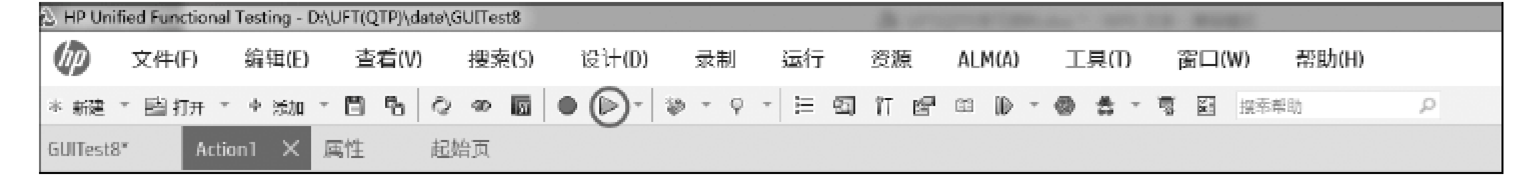


图 5.18　运行脚本

图 5.19 为 UFT 运行后的报告结果。

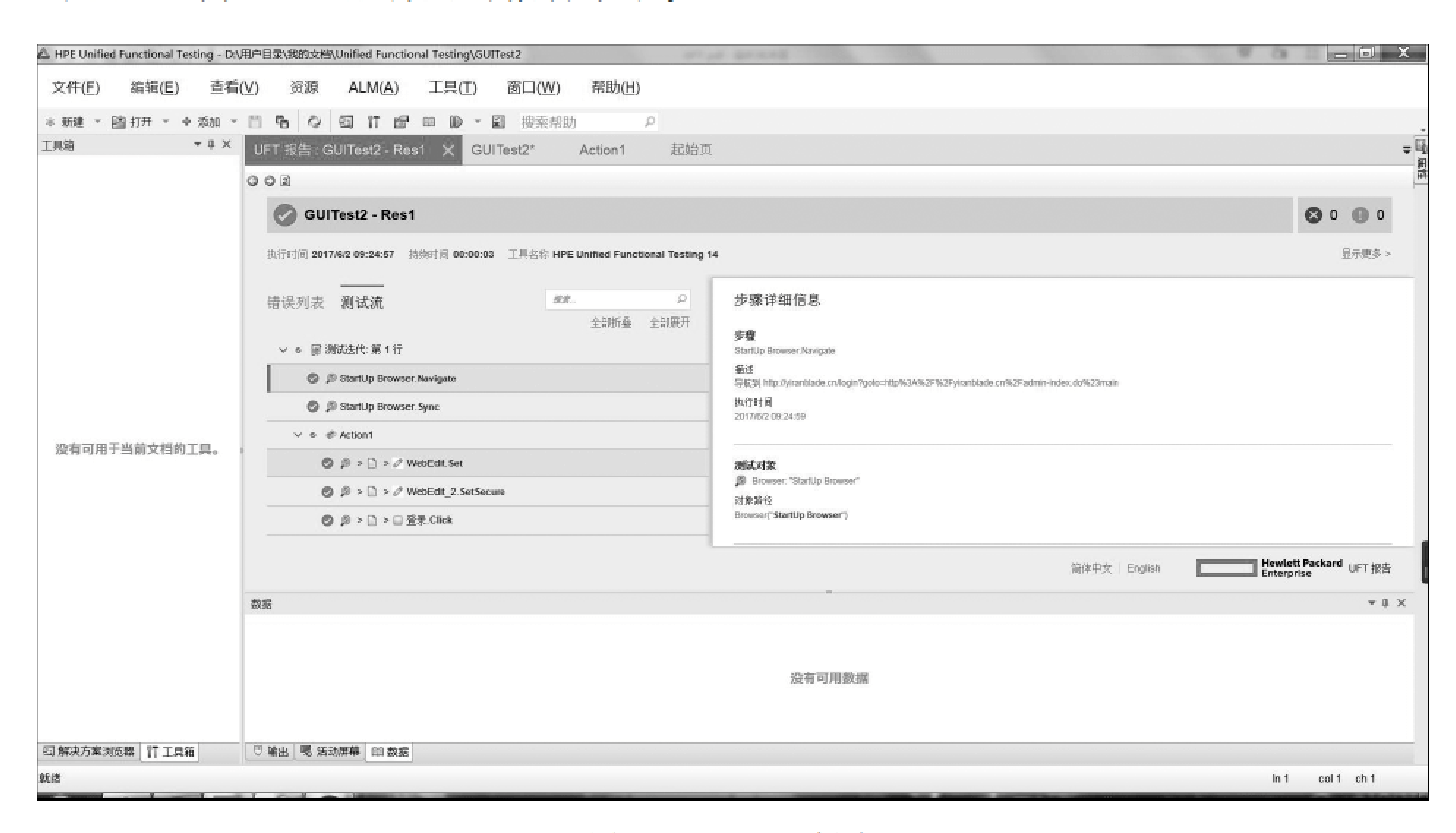


图 5.19　UFT 报告

2. UFT 参数化功能操作

录制或编辑测试脚本时，通过设置检查点的属性值，检查应用程序如何基于不同的数据执行相同的操作。UFT 参数化功能操作通过 DataTable 实现，可以使得测试数据在固定范围内变动，下面对前面的实验内容，即录制用户名和密码的脚本进行参数化设置。

步骤 1：选择“查看”菜单中的“关键字视图”命令，如图 5.20 所示。

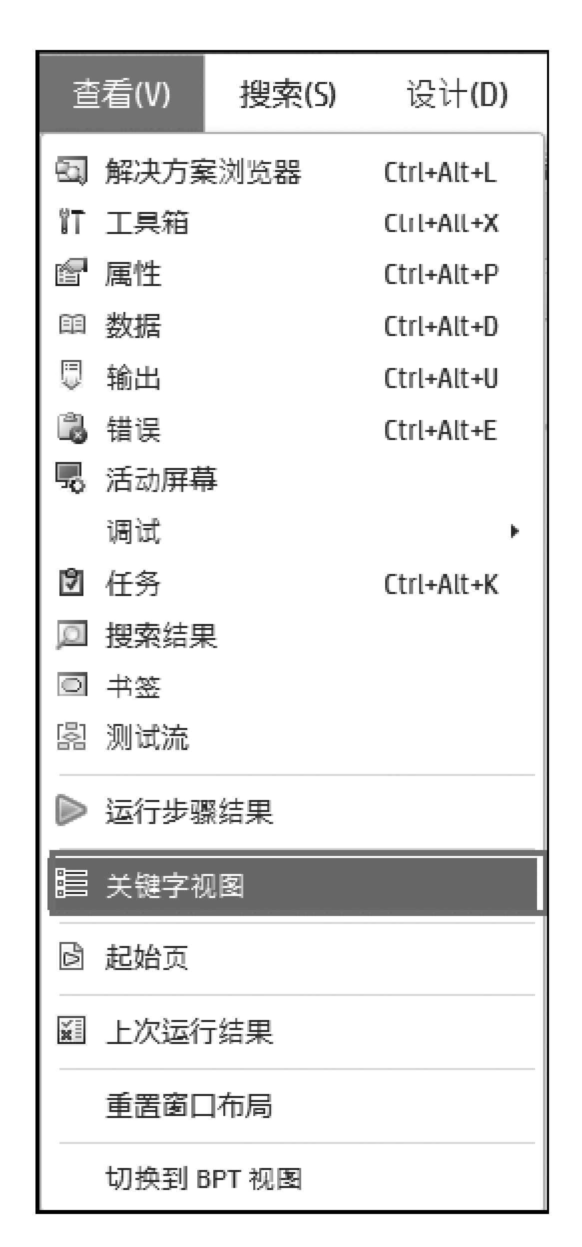


图 5.20　选择“关键字视图”命令

步骤 2：进入关键字视图，在 WebEdit 的“值”列单击＜＃p＞按钮，选择 DataTable (0)，单击“添加新参数”，如图 5.21 所示。

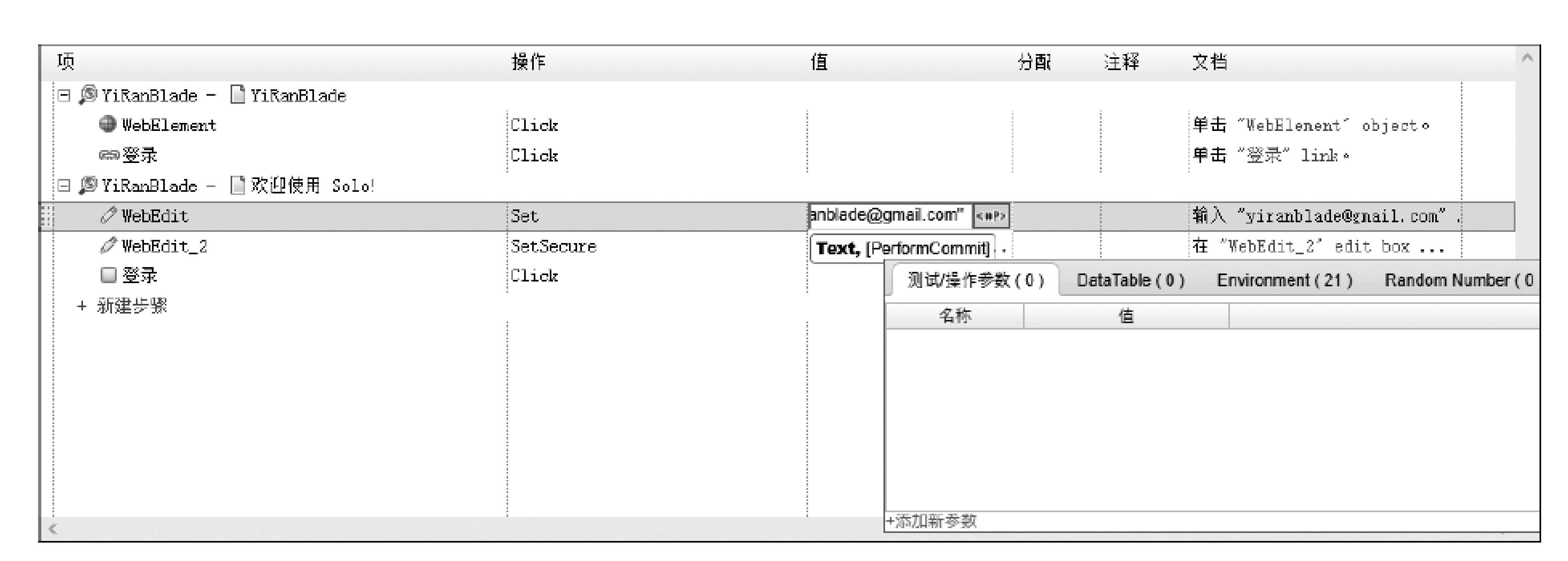


图 5.21　修改 DataTable 的内容

步骤 3：弹出“值配置选项”对话框，将“名称”修改为“用户名”，常量的值为"silence@creatshare.com"（值的内容包括双引号），如图 5.22 所示。

步骤 4：将“名称”修改为“密码”，常量的值不用修改，如图 5.23 所示。



图 5.22　将"名称"修改为"用户名"



图 5.23　将"名称"修改为"密码"

步骤 5：将用户名 yiranblade@gmail. com 改成 silence@yiranblade. com，修改方式如图 5.24 和图 5.25 所示。

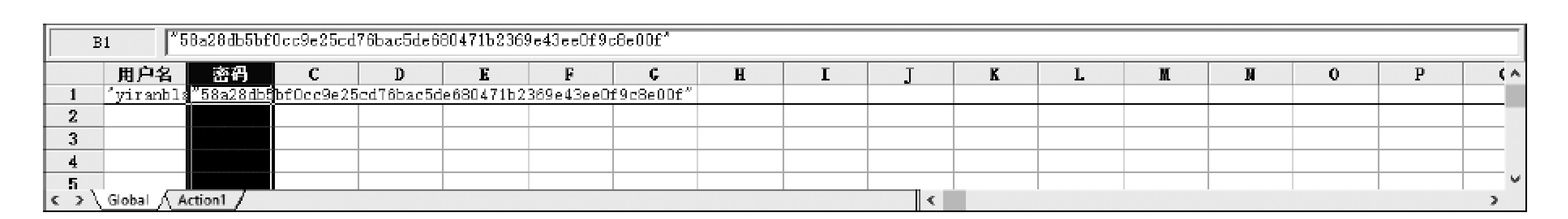


图 5.24　修改内容



图 5.25　选择"格式"→"自定义数字"命令

在图 5.24 中，选中"用户名"，右击，在快捷菜单中选择"格式"→"自定义数字"命令，如图 5.25 所示，在"单元格格式"对话框的"类型"列表中选择 0，如图 5.26 所示。

步骤 6：选择菜单"查看"→"编辑器"命令，如图 5.27 所示。可以看到，WebEdit 和 WebEdit_2 一栏中的代码变成"("用户名，dtGlobalSheet")"和"("密码，dtGlobalSheet")"，如图 5.28 所示。

在图 5.28 中的"用户名"和"密码"两列添加多个数据，UFT 每次取出一行数据，根据"用户名"和"密码"的内容进行回放，以此类推。



图 5.26　“单元格格式”对话框

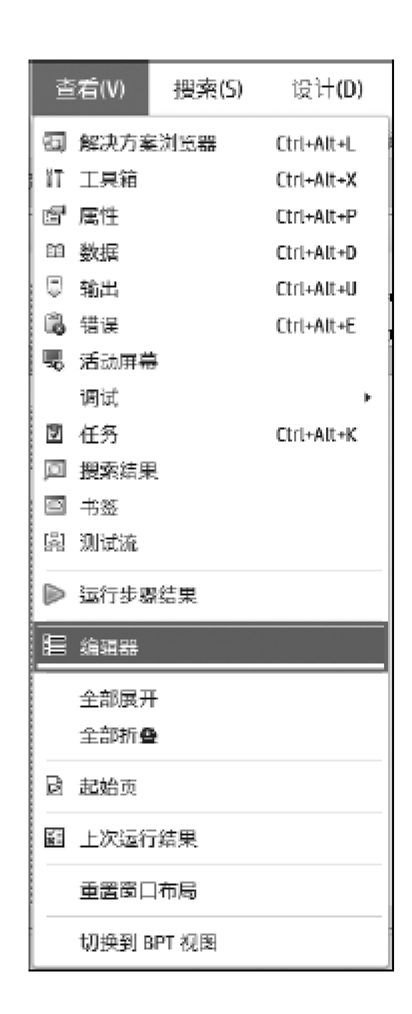


图 5.27　选择“编辑器”命令

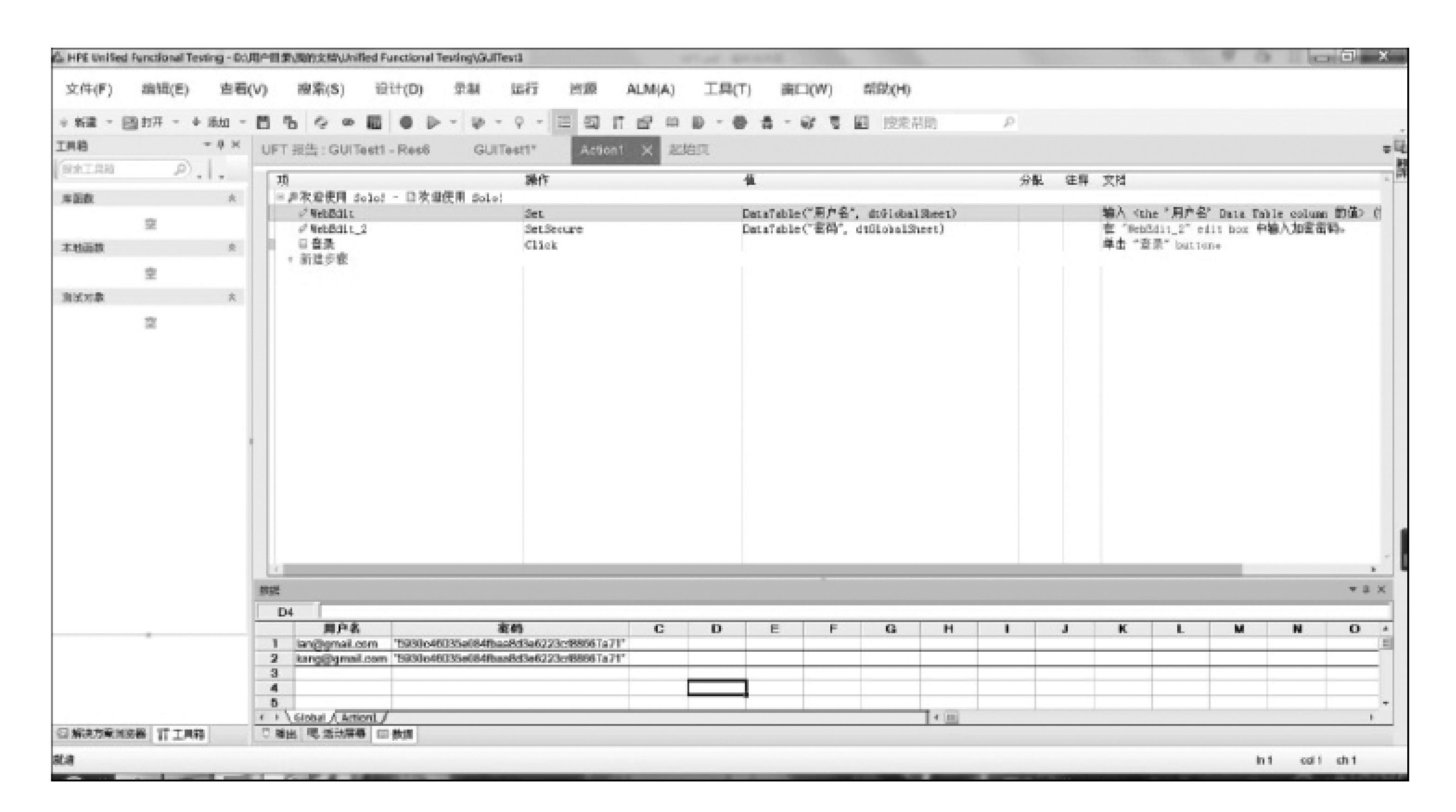


图 5.28　查看脚本



性能测试软件 LoadRunner

实验目的：

（1）理解 LoadRunner 的功能。

（2）熟练掌握 LoadRunner 的操作步骤。

实验环境：LoadRunner 软件。

6.1 LoadRunner 相关术语

Mercury LoadRunner 用于软件的负载测试。通过以模拟上千万的用户实施并发负载及实时性能监测的方式来确认和查找问题。使用 LoadRunner，能最大限度地缩短测试时间，优化性能，缩短应用系统的发布周期。

LoadRunner 相关术语如下：

- 虚拟用户生成器（VuGen）。用于捕获最终用户业务流程和创建自动性能测试脚本。VuGen 通过录制应用程序中典型最终用户执行的操作来生成虚拟用户。VuGen 将这些操作录制到自动虚拟用户脚本中，作为负载测试的基础。
- 控制器（Controller）。用于组织、驱动、管理和监控负载测试。控制器用来创建、管理和监控负载测试。使用控制器可以运行用来模拟真实用户执行的操作的脚本，并可以通过让多个虚拟用户同时执行这些操作来在系统中创建负载。
- 负载生成器。用于通过运行虚拟用户生成负载。
- Analysis。用于查看、分析和比较性能结果。Mercury Analysis 提供包含深入的性能分析信息的图和报告。使用这些图和报告，可以标识和确定应用程序中的瓶颈，并确定需要对系统进行哪些更改来提高系统性能。
- 场景。用于根据性能要求定义在每一个测试会话运行期间发生的事件。
- 虚拟用户（Vuser）。在场景中，LoadRunner 用虚拟用户代替实际用户。虚拟用户模拟实际用户的操作来使用应用程序。一个场景可以包含几十、几百甚至几千个虚拟用户。
- 事务。指服务器响应虚拟用户请求所用的时间，特点为：事务必须成对出现，即一个事务必须有开始（lr_start_transaction）和结束（lr_end_transaction）；事务结束函数包括两个参数，第一个参数是事务的名称，第二个参数是事务的状态；在应

用事务的过程中，不要将思考时间（lr_think_time 函数）放在事务开始和事务结束之间，否则思考时间将被计入事务的执行时间，从而影响对事务的正确执行时间的分析和统计。

6.2 LoadRunner 测试流程

LoadRunner 负载测试通常由 5 个步骤组成：计划、创建脚本、定义场景、运行场景和结果分析。

（1）计划。

定义性能测试要求，例如并发用户的数量、典型业务流程和所需响应时间。在任何类型的测试中，测试计划都是必要的步骤。测试计划是进行成功的负载测试的关键。任何类型的测试的第一步都是制定比较详细的测试计划。一个比较好的测试计划能够保证 LoadRunner 能够完成负载测试的目标。

（2）创建脚本。

LoadRunner 使用虚拟用户的活动来模拟真实用户操作 Web 应用程序，而虚拟用户的活动就包含在测试脚本中，开发测试脚本要使用 VuGen 组件。测试脚本要完成的内容如下：

- 每一个虚拟用户的活动。
- 定义结合点。
- 定义事务。

（3）定义场景。

使用 LoadRunner 控制器设置负载测试环境。

（4）运行场景。

通过 LoadRunner 控制器驱动、管理和监控负载测试。

（5）结果分析。

使用 LoadRunner Analysis 创建图和报告并评估性能。

6.3 实验步骤

本实验使用的是 LoadRunner 8.1 版本，LoadRunner 12 版本之后 VuGen、Controller、Analysis 分为 3 个客户端使用。

LoadRunner 8.1 自带 Flight Reservation（预订航班）系统作为测试示例。预订航班系统是一个基于 Web 的旅行代理应用程序，本测试要确定多个用户同时执行相同的事务时该应用程序将如何处理。使用 LoadRunner 代替旅行代理，可以创建具有 1000 个虚拟用户的场景，并且这些虚拟用户可以同时尝试在应用程序中预订航班。

LoadRunner 测试过程由以下 4 个基本步骤组成。

步骤 1：创建脚本。捕获在应用程序中执行的典型最终用户业务流程。

步骤 2：设计场景。通过定义测试会话期间发生的事件，设置负载测试环境。

步骤 3：运行场景。运行、管理并监控负载测试。

步骤 4：分析结果。分析负载测试期间 LoadRunner 生成的性能数据。

6.3.1 使用 VuGen 创建脚本

创建负载测试的第一步是使用 VuGen 录制典型最终用户的业务流程。VuGen 采用录制-回放机制。当在应用程序中按照业务流程操作时，VuGen 将这些操作录制到自动脚本中，以便作为负载测试的基础。

在本节中，将录制旅行代理为一位乘客预订从丹佛到洛杉矶的航班的流程。

1. 准备录制

打开 VuGen 并创建一个空白脚本。

(1) 启动 LoadRunner。

选择“开始”→“程序”→Mercury LoadRunner→LoadRunner 命令，将打开 Mercury LoadRunner 窗口，如图 6.1 所示。

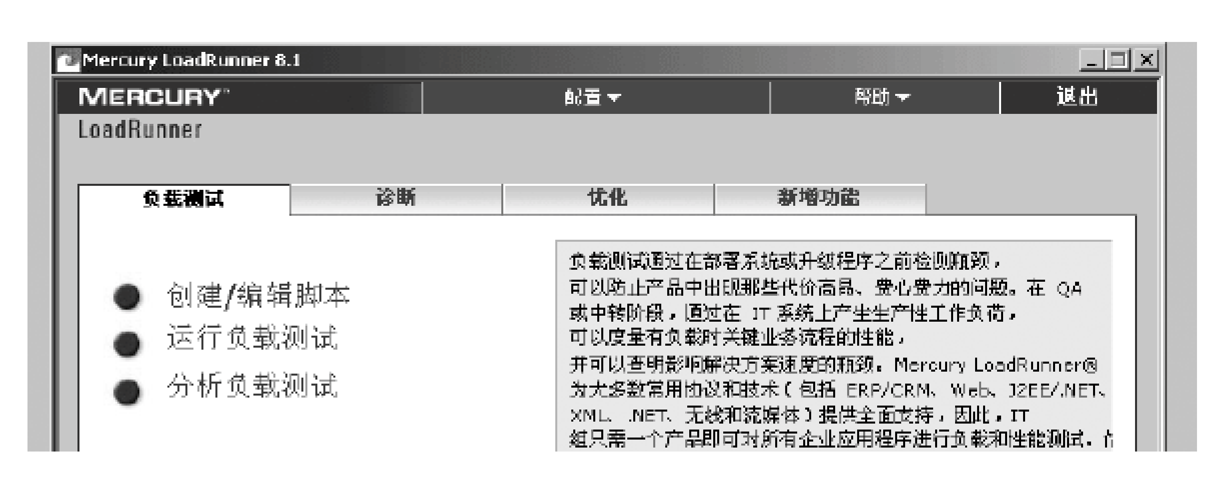


图 6.1 LoadRunner 开始界面

(2) 打开 VuGen。

在“负载测试”选项卡中，单击“创建/编辑脚本”，打开 VuGen 的起始页，如图 6.2 所示。

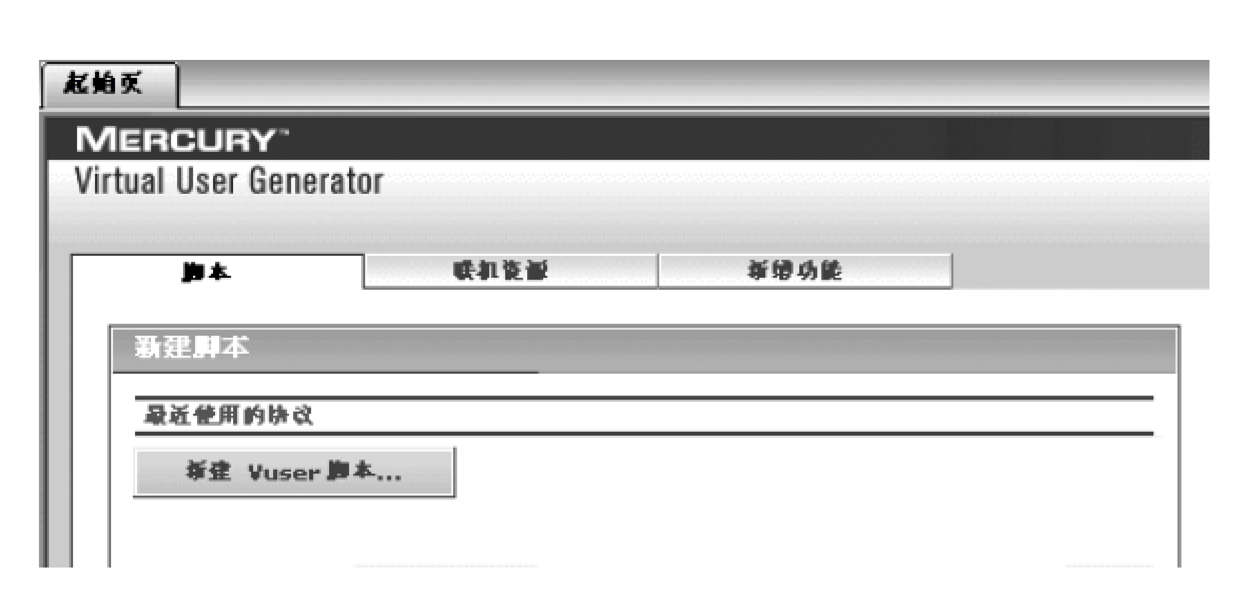


图 6.2 VuGen 界面

(3) 创建一个空白的 Web 脚本。

在 VuGen 的起始页的“脚本”选项卡中，单击 “新建 Vuser 脚本”按钮，打开“新建虚拟用户”对话框，并显示“新建单协议脚本”，如图 6.3 所示。

确保“类别”类型为 All Protocols(所有协议)。VuGen 将显示所有可用于单协议脚本的协议列表。向下滚动查看该列表，选择 Web(HTTP/HTML)，并单击“确定”按钮创



图 6.3 LoadRunner 新建虚拟用户界面

建一个空白 Web 脚本。

2. 使用 VuGen 向导录制业务流程

空脚本以 VuGen 的向导模式打开,且任务窗格显示于左侧(如果未显示任务窗格,请单击工具栏上的"任务"按钮)。VuGen 的向导将指导用户逐步完成脚本创建,然后根据用户的测试环境进行相应的修改。任务窗格列出了脚本创建过程中的每个步骤或任务。在逐步完成每一步操作的过程中,VuGen 会在窗口的主区域显示详细的说明和准则,如图 6.4 所示。

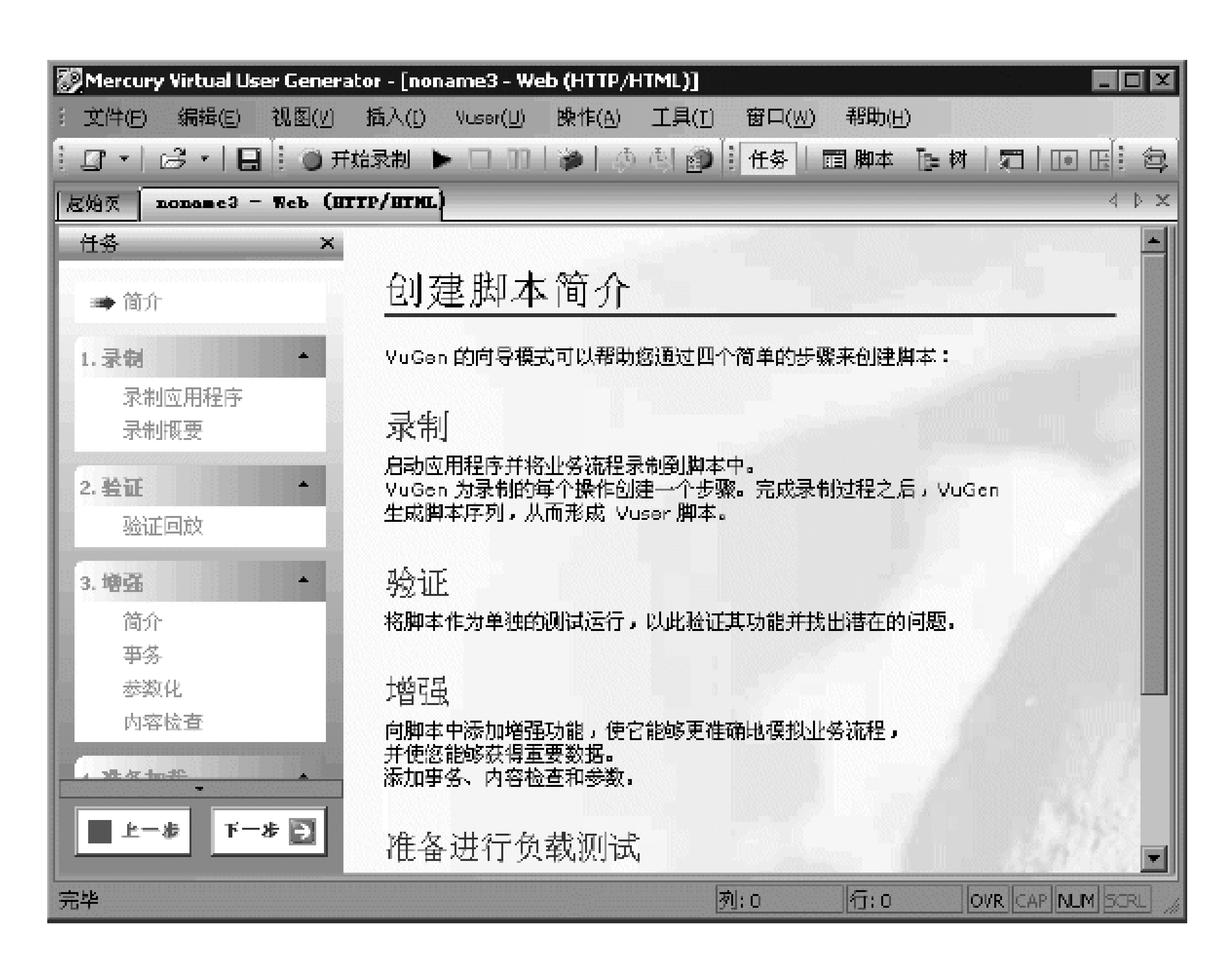


图 6.4 VuGen 界面

录制脚本的具体过程如下:

(1) 在 Mercury Tours 网站上开始录制。

在任务窗格中,单击"开始录制",打开"开始录制"对话框,如图 6.5 所示。

在"URL 地址"栏中,输入 http://127.0.0.1:1080/WebTours/index.htm。在"录制

到操作”框中选择 Action，单击“确定”按钮，打开一个新的 Web 浏览器，并显示 Mercury Tours 站点(如果在打开站点时出现错误，请确保 Web 服务器正在运行。要启动服务器，请选择“开始”→“程序”→Mercury LoadRunner→“示例”→Web→“启动 Web 服务器”)。同时会打开浮动的录制工具栏，如图 6.6 所示。




图 6.5　VuGen 录制对话框

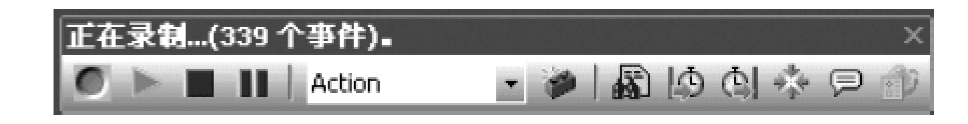


图 6.6　录制工具栏

(2) 登录 Mercury Tours 网站。

输入账号 jojo 和密码 bean，单击“登录”按钮，打开欢迎页面。

(3) 输入航班详细信息。

单击“航班”，打开“查找航班”页，设置航班信息。

- 出发城市：丹佛(默认设置)。
- 出发日期：保持默认设置不变(当前日期)。
- 到达城市：洛杉矶。
- 返回日期：保持默认设置不变(第二天的日期)。

保持其余的默认设置不变，然后单击“继续”按钮，打开“搜索结果”页。

(4) 选择航班。

单击“继续”按钮，接受默认航班选择，将打开“付费详细信息”页。

(5) 输入付费信息并预订航班。

单击“继续”按钮，打开“发票”页，并显示发票。

(6) 查看路线。

在左窗格中单击“路线”，将打开“路线”页。

(7) 在左窗格中单击“注销”。

(8) 单击浮动工具栏上的停止按钮以停止录制过程。

一旦生成了虚拟用户脚本，虚拟用户向导将自动前进到任务窗格中的下一步，并显示包含协议信息以及在会话期间创建的一系列操作的录制概要。对于录制期间执行的每个步骤，VuGen 都生成一个快照(即录制期间各窗口的图片)。这些快照的缩略图显示在右侧窗格中。

(9) 选择“文件”→“保存”命令，或单击“保存”按钮。在“文件名”框中输入 basic_tutorial。单击“保存”按钮，VuGen 将把该文件保存在 LoadRunner 脚本文件夹中，并在标题栏中显示该测试名称。

3. 查看脚本

现在,可在树视图或脚本视图中查看脚本。树视图是基于图标的视图,其中将虚拟用户的操作作为步骤列出;而脚本视图是基于文本的视图,其中将虚拟用户的操作作为函数列出。

1）树视图

在树视图中查看脚本,可选择“查看”→“树视图”命令或单击“树视图”按钮。对于录制 期间执行的每个步骤,VuGen 都在测试树中生成了一个图标和一个标题,如图 6.7 所示。



图 6.7　树视图

在树视图中,将用户的操作作为脚本步骤列出。大多数步骤都附带相应的录制快照。

2）脚本视图

脚本视图是基于文本的视图,其中将虚拟用户的操作作为 API 函数列出。选择“查看”→“脚本视图”命令或单击“脚本视图”按钮打开脚本视图,如图 6.8 所示。

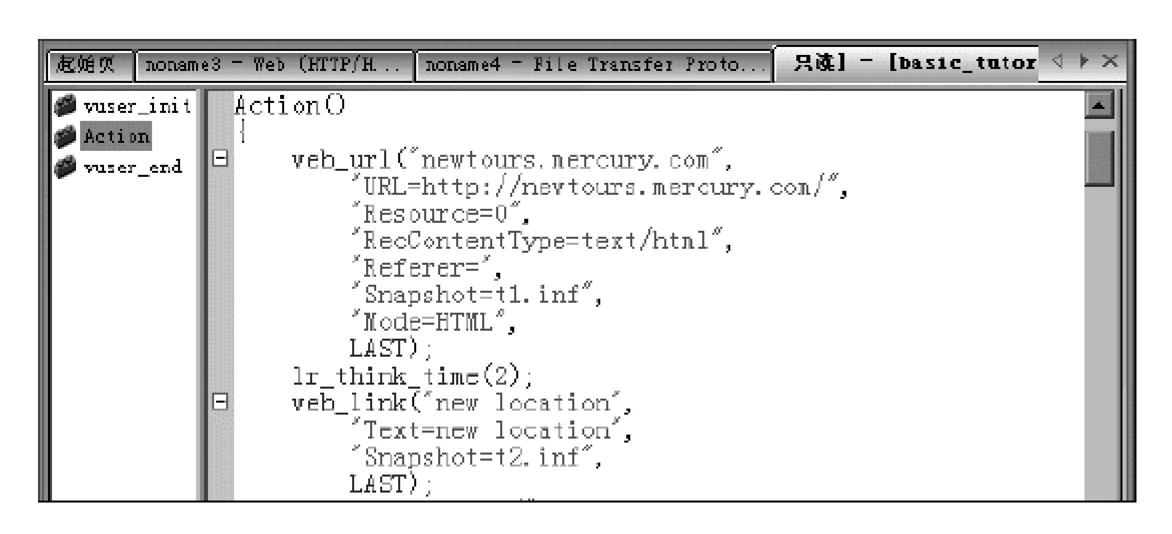


图 6.8　脚本视图

在脚本视图中,VuGen 在编辑器中通过彩色编码函数及其参数值显示脚本。可以直接在此窗口输入 C 语言或 LoadRunner API 函数以及控制流语句。

4. 回放脚本

完成录制后,就可以回放脚本,以便验证它是否准确地模拟了录制的操作。

(1) 确保显示了任务窗格(如果没有,请单击工具栏中的“任务”按钮)。单击任务窗格中的“验证回放”,然后单击说明窗格底部的“开始回放”按钮。

(2) 如果打开了“选择结果目录”对话框,询问要存储结果目录的位置,请接受默认名称并单击“确定”按钮。

(3) 单击任务窗格中的“验证回放”查看回放概要。回放概要列出了可能检测到的所有错误并显示录制和回放快照的缩略图,可以通过“运行时设置”模拟各种不同类型的用户行为。例如,可以模拟一个对服务器立即做出响应的用户,也可以模拟一个在做出响应之前先停下来思考的用户。

5. 增强脚本

准备负载测试过程时,LoadRunner 允许用户增强脚本以使其更好地反映真实情况。例如,可以在脚本中插入名为内容检查的步骤,以验证某些特定内容是否显示在返回页上。可以修改脚本来模拟多用户行为,也可以指示 VuGen 度量特定的业务流程。

准备要部署的应用程序时,需要度量特定业务流程的持续时间,如登录、预订航班等花费的时间。这些业务流程通常由脚本中的一个或多个步骤或操作构成。在 LoadRunner 中,可以通过将想要度量的操作标记为事务来指定一系列操作。

下面在脚本中插入一个事务以度量用户查找和确认航班所花费的时间。

(1) 打开事务创建向导。

确保显示了任务窗格(如果没有,请单击“任务”按钮)。在任务窗格的“增强功能”标题下,单击“事务”,打开事务创建向导。事务创建向导显示脚本中不同步骤的缩略图。

单击“新建事务”按钮。现在,可以拖动事务标记并将其放置在脚本中的指定点,向导会提示插入事务的起始点,如图 6.9 所示。

事务

名称	TPS	通过	失败	停止
book_flight	0.0	4	0	0
search_flights	0.0	4	0	0
vuser_end_Transaction	0.0	2	0	0
vuser_init_Transaction	0.0	2	0	0
BookFlight_Transaction_Transaction	0.0	4	0	0
logon	0.0	4	0	0

关闭(C)　帮助(H)

图 6.9　新建事务

(2) 插入开始事务标记和结束事务标记。

拖动标记并放置到标题为“搜索航班按钮”的第三个缩略图之前并单击,向导会提示插入结束点。

(3) 指定事务的名称。

向导提示输入事务的名称。输入 find_confirm_flight,通过将标记拖动到脚本中的其他点来调整事务的起始点或结束点。

6.3.2 使用 Controller 设计和运行场景

使用控制器,可以将应用程序性能测试需求划分为多个场景。场景定义每个测试会话中发生的事件。例如,一个场景可以定义和控制模拟的用户数、用户执行的操作以及用

户运行其模拟时所用的计算机。

1）创建场景

此部分的目标是创建一个场景，用来模拟10个旅行代理同时登录系统、搜索航班、购买机票、查看路线和注销系统。

（1）打开 Mercury LoadRunner。

选择“开始”→“程序”→Mercury LoadRunner→LoadRunner 命令，打开 Mercury LoadRunner 窗口。

（2）打开控制器。

在“负载测试”选项卡中，单击“运行负载测试”，打开 LoadRunner 控制器。默认情况下，控制器打开时将显示“新建场景”对话框，如图6.10所示。

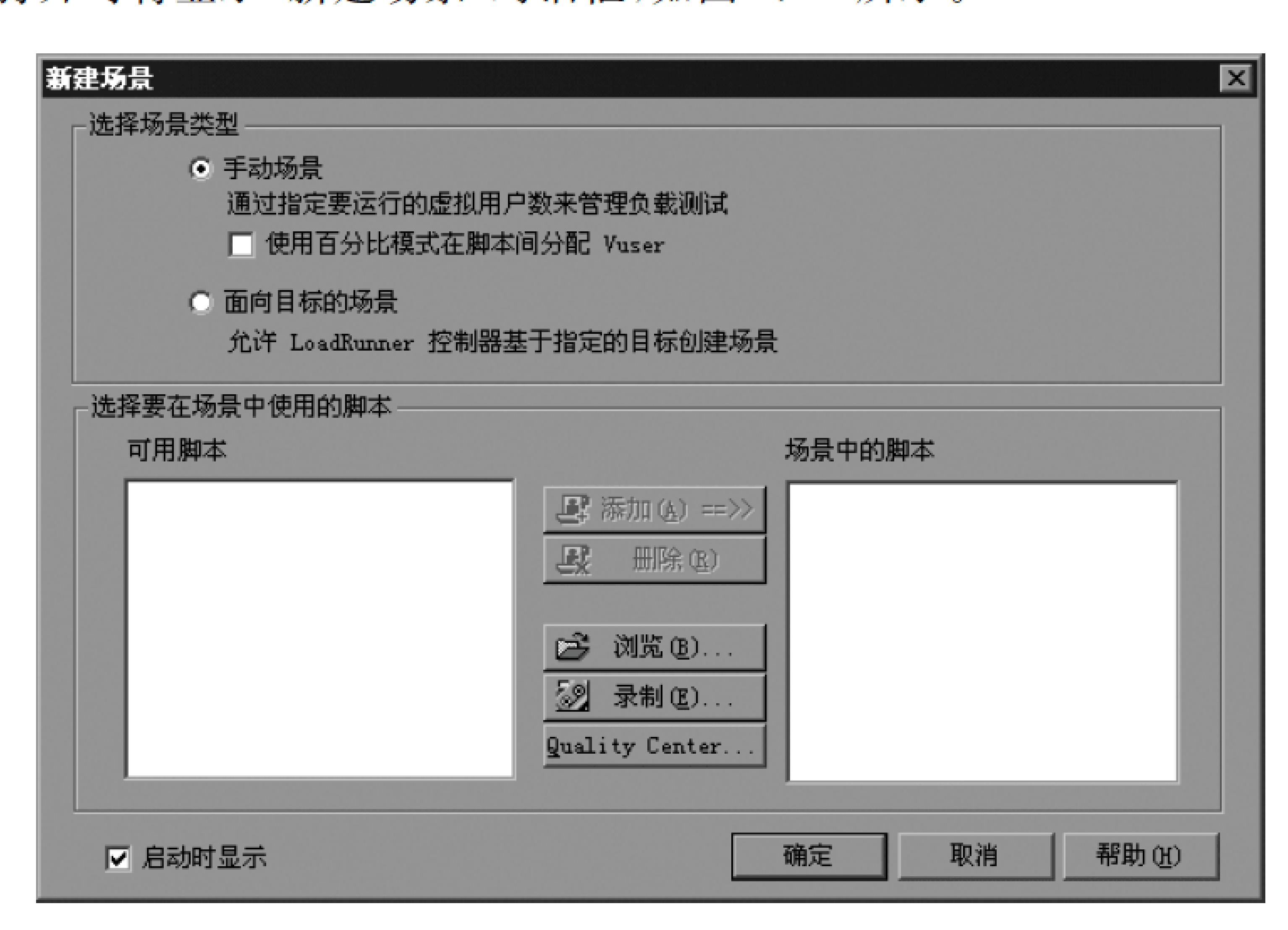


图6.10 “新建场景”对话框

（3）选择场景类型。

选择“手动场景”。控制器允许选择各种不同的场景类型。

（4）向负载测试添加脚本。

单击“浏览”按钮，找到 LoadRunner 安装文件夹下 Tutorial 目录中的 basic_script。“可用脚本”部分和“场景中的脚本”部分中将显示该脚本。单击“确定”按钮。LoadRunner 控制器的“设计”选项卡中将显示创建的场景。

2）设计场景

控制器窗口的“设计”选项卡包含“场景计划”和“场景组”两个主要部分，如图6.11所示。

（1）场景计划。在该部分，可以设置负载行为以准确描绘用户行为。可以确定将负载应用于应用程序的频率、负载测试持续时间和停止负载的方式。

（2）场景组。可以在该部分配置虚拟用户组。可以创建不同的组来代表系统的典型用户。可以定义这些典型用户运行的操作、运行的虚拟用户数以及虚拟用户运行时所用

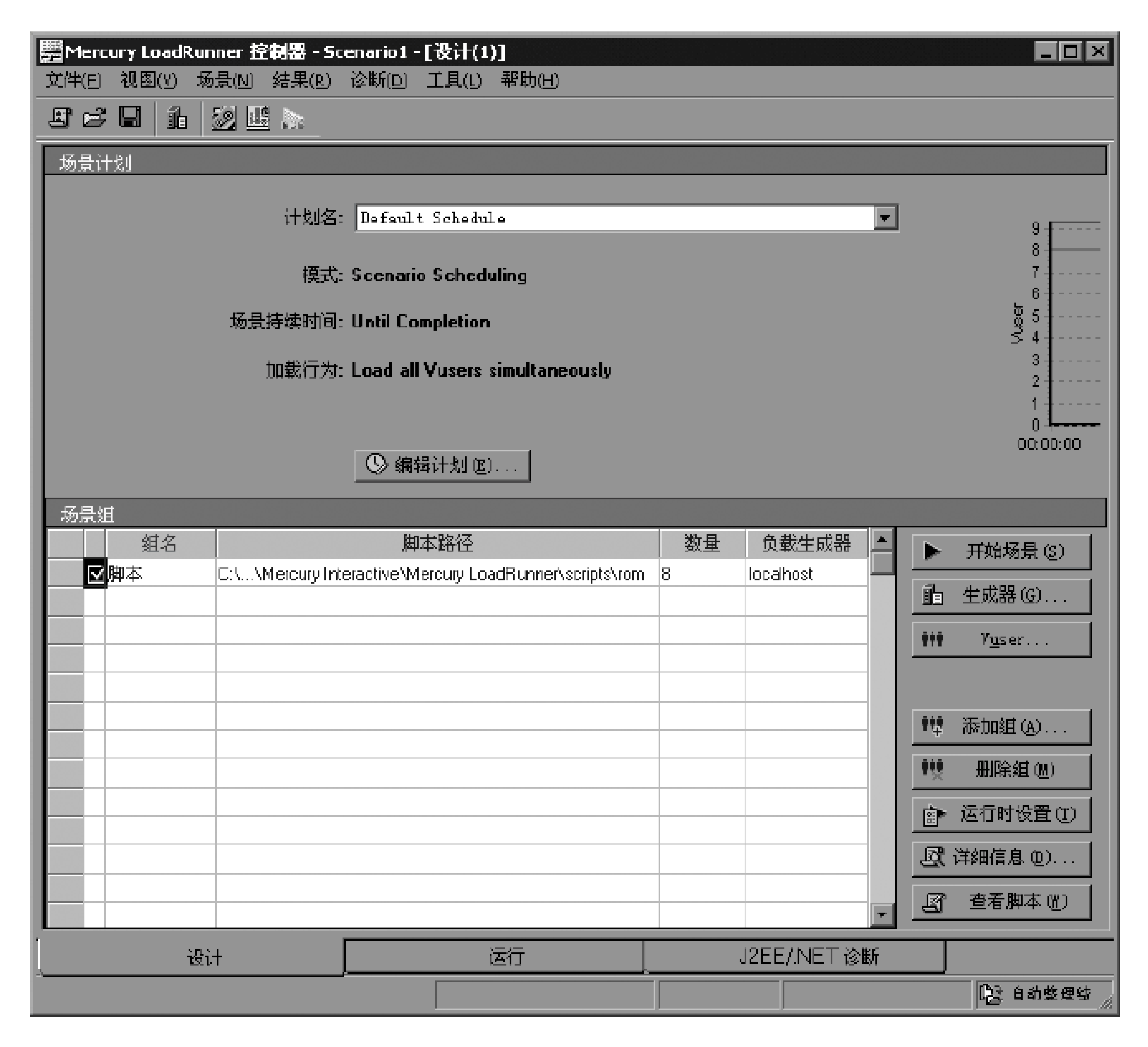


图 6.11　控制器窗口的“设计”选项卡

的计算机。

（3）负载生成器。是通过运行虚拟用户在应用程序中创建负载的计算机。可以使用多台负载生成器计算机，并在每台计算机上创建许多虚拟用户。

3）计划场景

由于通常不会有多个典型用户恰好同时登录和注销系统，所以 LoadRunner 控制器的计划生成器允许建立较准确地描绘典型用户行为的场景计划。例如，在创建手动场景后，设置场景的持续时间或选择在场景中逐渐运行和停止虚拟用户。

下面，使用控制器的计划生成器更改默认负载设置。

（1）更改场景计划默认设置。

单击“编辑计划”按钮，将打开计划生成器，如图 6.12 所示。

（2）指定逐渐开始。

在“加压”选项卡中，将设置更改为每 15 秒开始 2 个。

（3）计划持续时间。

在“持续时间”选项卡中，将设置更改为在加压完成之后运行 3 分钟。

（4）计划逐渐关闭。

在“减压”选项卡中，将设置更改为每 30 秒停止 5 个。单击“确定”按钮。

完成了负载测试场景的设计，接下来运行该测试并观察应用程序如何在负载下运行。在开始运行测试之前，应该先熟悉控制器窗口的“运行”选项卡。“运行”选项卡是管理和监控测试的控制中心。

单击“运行”选项卡打开“运行”视图，如图 6.13 所示。

“运行”视图包含 5 个主要部分：



图 6.12　计划生成器



图 6.13　打开“运行”视图

(1) 场景组。位于左上窗格中，可以查看场景组中的虚拟用户的状态。使用该窗格右侧的按钮可以启动、停止和重置场景，查看单个虚拟用户的状态，并且可以手动添加更多的虚拟用户，从而增加场景运行期间应用程序上的负载。

(2) 场景状态。位于右上窗格中，可以查看负载测试的概要，其中包括正在运行的虚拟用户数以及每个虚拟用户操作的状态。

(3) 可用图树。位于中部左侧窗格中，可以查看 LoadRunner 的图列表。若要打开图，请在该树中选择一个图，然后将其拖动到图查看区域中。

(4) 图查看区域。位于中部右侧窗格中，查看 1～8 个图(选择“视图”→“查看图”

命令）。

（5）图例。位于底部窗格中，可以查看选定的图中的数据。

运行步骤如下：

（1）开始场景。

单击“开始场景”按钮开始运行测试。控制器开始运行场景。

（2）通过控制器的联机图监控性能。

测试运行时，通过 LoadRunner 的集成监控器查看应用程序如何实时执行以及潜在瓶颈所在位置，也可在控制器的联机图上查看监控器收集的性能数据。联机图显示在“运行”选项卡的图查看区域。

6.3.3 使用 Analysis 分析场景结果

使用 LoadRunner Analysis 分析场景运行期间生成的性能数据，将性能数据收集到详细的图和报告中，确定和标识应用程序中的瓶颈以及提高系统性能所需的改进。

下面提供了一个 Analysis 会话示例，该会话所基于的场景与前面运行的场景相似。

（1）从控制器的菜单中选择“工具”→Analysis 命令或选择“开始”→“程序”→Mercury LoadRunner→“应用程序”→Analysis 命令打开 LoadRunner Analysis。

（2）在 Analysis 窗口中，选择“文件”→“打开”命令，将打开“打开现有 Analysis 会话文件”对话框。

（3）在 LoadRunner 安装目录下的 Tutorial 文件夹中选择 analysis_session，将在 Analysis 窗口中打开该会话文件。

- 概要报告。LoadRunner Analysis 打开时显示概要报告。概要报告提供有关场景运行的一般信息。在报告的统计信息概要中，可以了解到测试中运行的用户数，查看其他统计信息（如总 / 平均吞吐量和总 / 平均点击次数）。报告的事务概要列出了每个事务的行为概要。
- 查看图。Analysis 窗口左窗格的图树中列出了已经打开可供查看的图。这些图显示在 Analysis 窗口右窗格的图查看区域中。

（4）通过平均事务响应时间图可以查看在场景中运行的每一秒中有问题的事务行为。

① 在图树中单击“平均事务响应时间”，平均事务响应时间图即显示在图查看区域中。

② 单击 check_itinerary，如图 6.14 所示。

（5）合并图和关联图。将两个图联系起来，就会看到一个图的数据会对另一个图的数据产生影响，这称为将两个图关联。例如，可以将正在运行的 Vuser 图和平均事务响应时间图相关联，来了解大量的虚拟用户对事务的平均响应时间产生的影响。

① 在图树中单击“正在运行的 Vuser”，查看正在运行的 Vuser 图。

② 右击正在运行的 Vuser 图并选择“合并图”命令。

③ 在“选择要合并的图”列表中，选择“平均事务响应时间”。

④ 在“选择合并类型”区域中，选择“关联”，然后单击“确定”按钮。

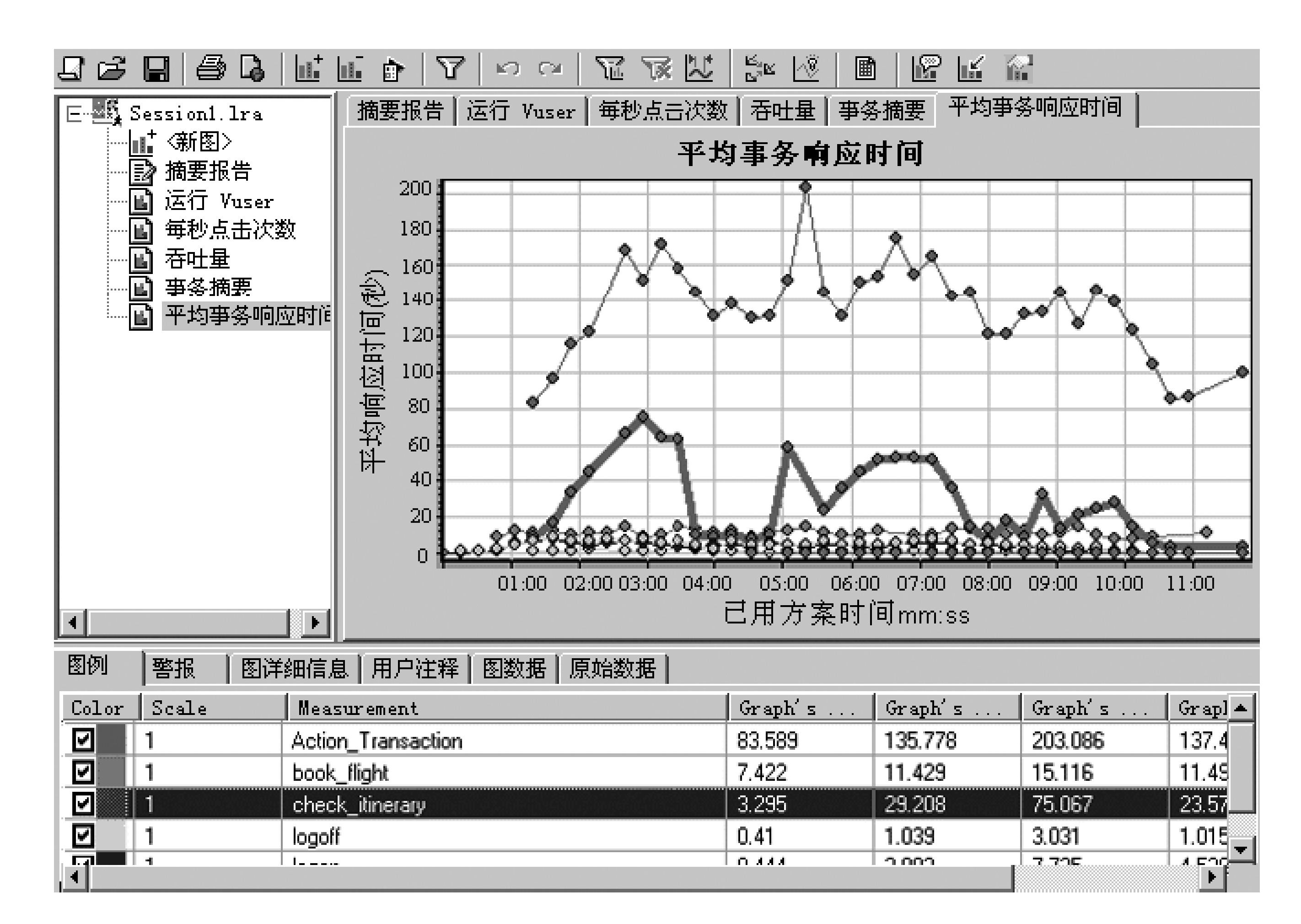


图 6.14　查看图表

Vusers 和平均事务响应时间如图 6.15 所示。

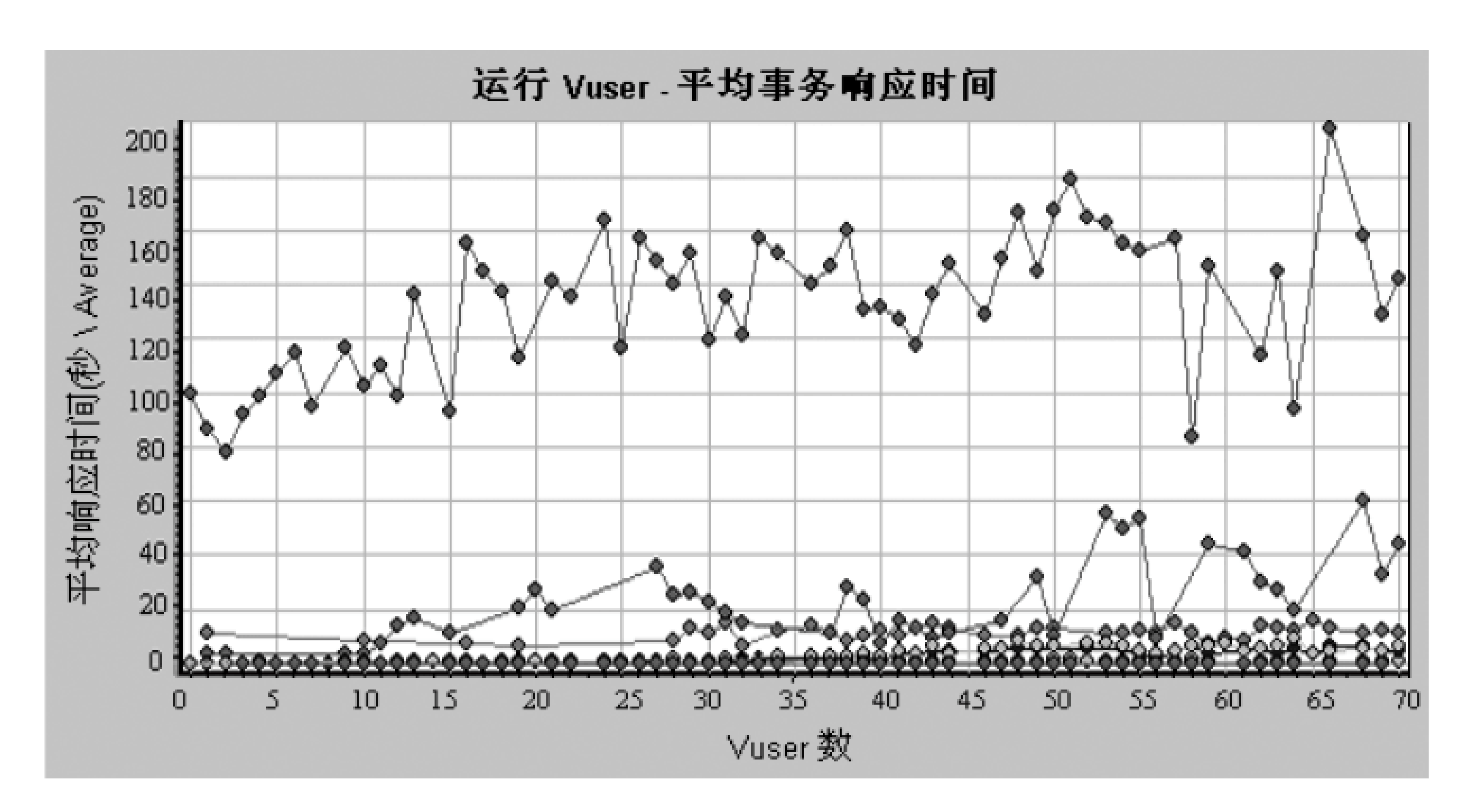


图 6.15　Vusers 和平均事务响应时间

另一个 Analysis 工具自动关联用来合并所有包含可能已对给定事务产生影响的数据的图，事务与每个元素的关联都会显示出来。

(6) 筛选图数据和排序图数据。

对图数据进行筛选，显示特定场景段的较少事务。对图数据进行排序，以更多的相关方式来显示数据。

① 在图树中单击“平均事务响应时间”打开该图。

② 右击该图并选择“设置筛选器”→“分组方式”。

③ 在“事务名称”值框中，选择 check_itinerary 并单击“确定”。筛选的图仅显示 check_itinerary 事务，而隐藏所有其他事务。

(7) 发布 HTML 报告或 Microsoft Word 报告。

采用 HTML 报告或 Microsoft Word 报告的形式发布 Analysis 会话的结果。



代码分析工具 FindBugs

实验目的：

(1) 理解 FindBugs 的功能。

(2) 熟练掌握 FindBugs 的操作步骤。

实验环境：FindBugs 软件。

7.1 FindBugs 简介

FindBugs 是一个静态分析工具，将字节码与一组缺陷模式进行对比，在不实际运行程序的情况下对软件进行分析。FindBugs 对.class 文件及其源文件(.java 文件)进行分析，可以定位到出现问题的代码。FindBugs 可以发现许多代码中间潜在的缺陷，如引用了空指针、特定的资源未关闭等。

7.2 实验内容

7.2.1 安装 FindBugs

在 Eclipse 中安装 FindBugs 的步骤如下。

步骤 1：打开 Eclipse，选择 Help→Install New Software 命令，如图 7.1 所示。

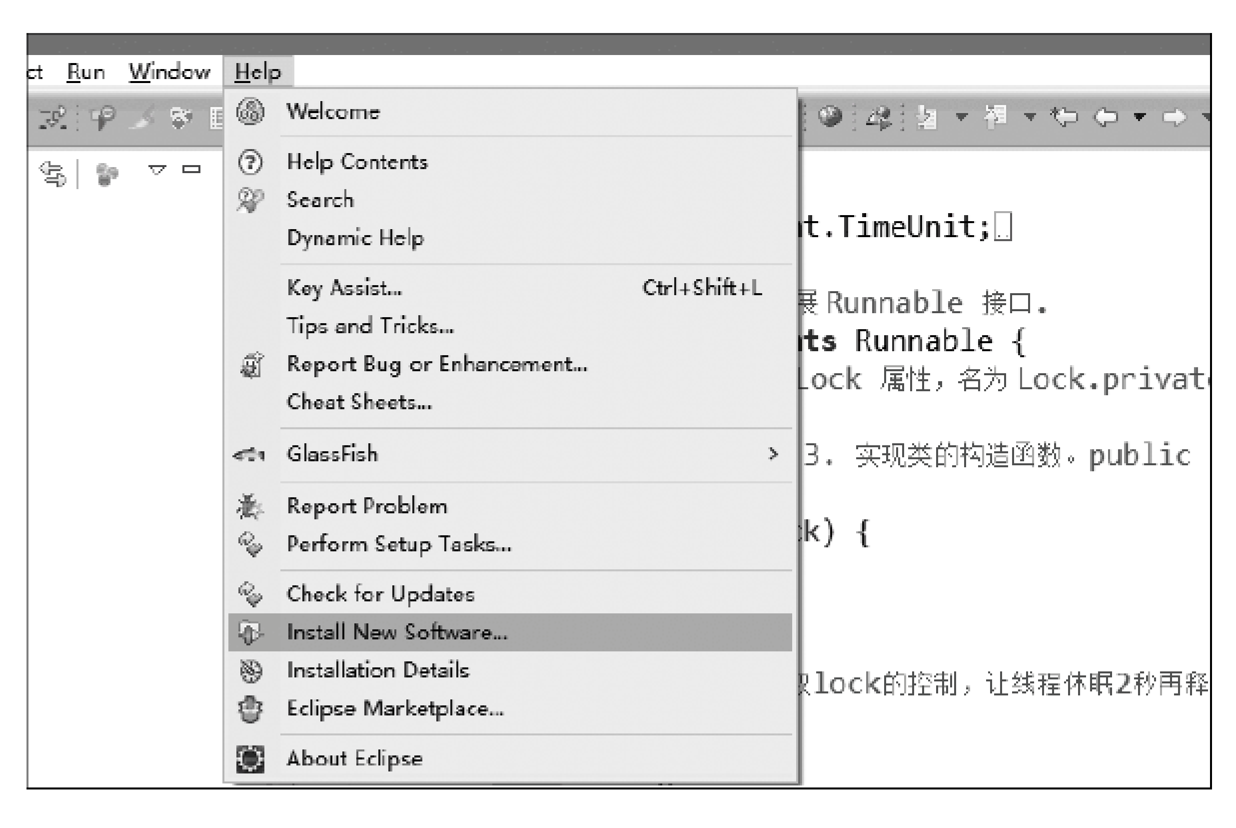


图 7.1 安装 FindBugs

步骤 2：在地址栏输入 http://findbugs.cs.umd.edu/eclipse，如图 7.2 所示。

图 7.2　安装 FindBugs

步骤 3：勾选 FindBugs，然后依次单击 next→next→finish 按钮。

步骤 4：重启 Eclipse，插件安装成功。

7.2.2　FindBugs 使用方法

被测试文件 Task.java 内容如下：

```
import java.util.concurrent.TimeUnit;
import java.util.concurrent.locks.ReentrantLock;
//1. 创建一个类,名为 Task,扩展 Runnable 接口
public class Task implements Runnable {
    //2. 声明一个 ReentrantLock 属性,名为 Lock.private
    ReentrantLock lock;//3. 实现类的构造函数。public
    Task(ReentrantLock lock){
        this.lock=lock;
    }
    //4. 实现 run()方法。获取 lock 的控制,让线程休眠 2 秒再释放 lock
    @Override
    public void run(){
        lock.lock();
        try {
            TimeUnit.SECONDS.sleep(1);
            lock.unlock();
        } catch(InterruptedException e){
            e.printStackTrace();
        }
    }
    //5.创建例子的主类,名为 Main 并添加 main()方法
    public static class Main {
        public static void main(String[] args){
            //6. 声明并创建一个 ReentrantLock 对象,名为 lock
```

```
            ReentrantLock lock=new ReentrantLock();
            //7.创建 10 个 Task 对象和 10 个线程来执行这些任务。调用 run()方法开始线程
            for(int i=0; i<10; i++){
                Task task=new Task(lock);
                Thread thread=new Thread(task);
                thread.run();
            }
        }
    }
}
```

右击被测文件,在快捷菜单中选择 Find Bugs→Find Bugs 命令,就会扫描该文件,检查哪些代码有问题,如图 7.3 所示。

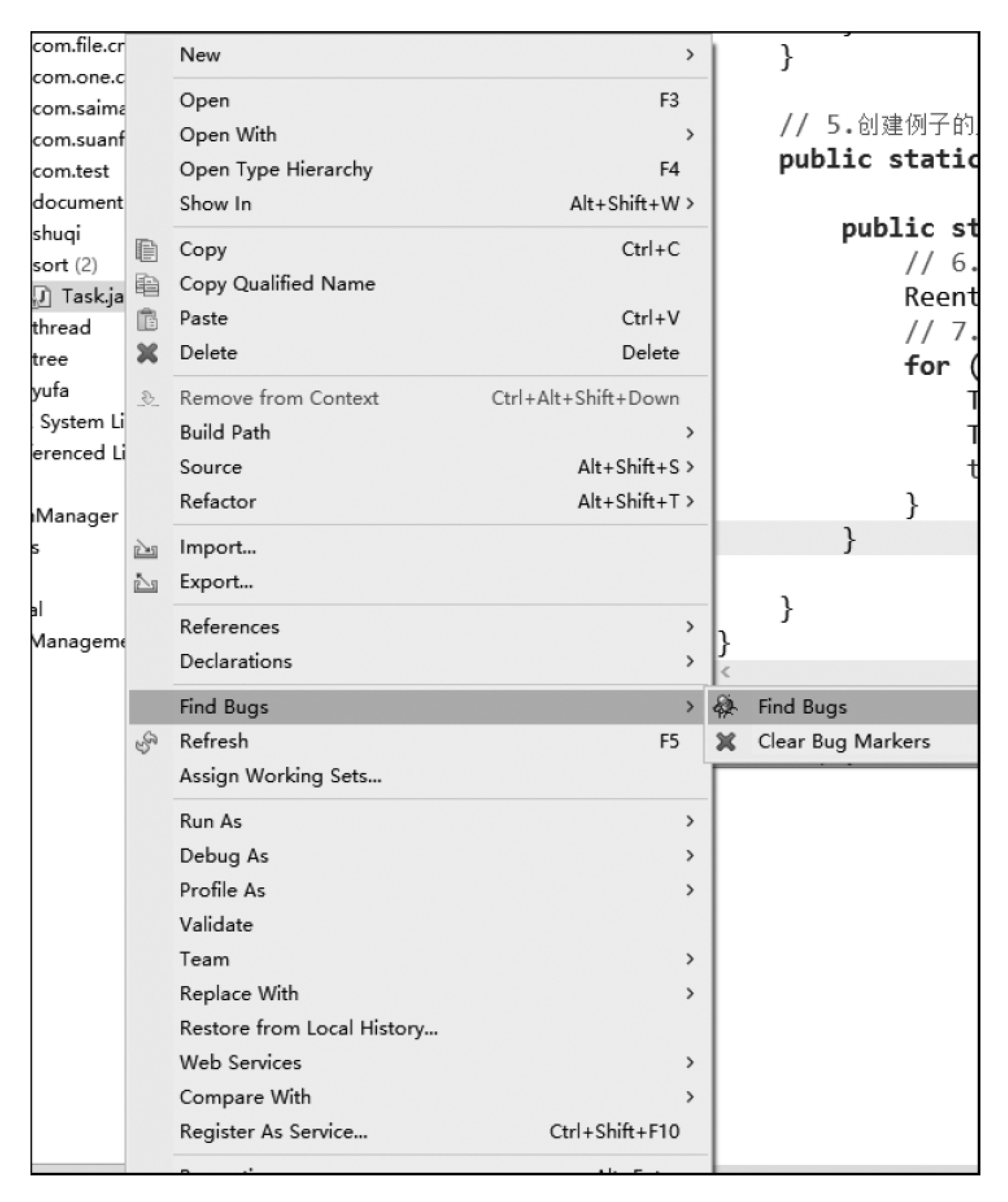


图 7.3 扫描代码

在 Bug Explorer 中右击一个代码问题(左侧有一个小虫子图标),在快捷菜单中选择 properties 命令查看详细信息,如图 7.4 所示。

FindBugs 运行后在源代码编辑器中产生的警告标识如图 7.5 所示。

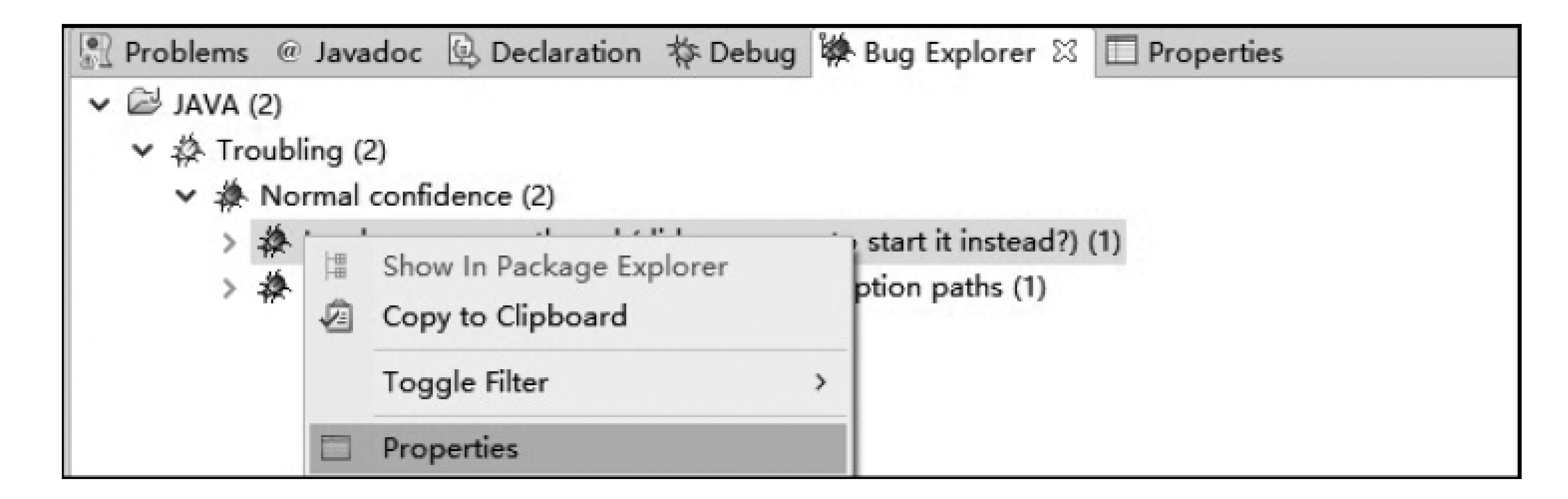


图 7.4　扫描出来的代码问题

```
// 4. 实现run() 方法，获取lock的控制，让线程休眠2秒再释放lock
@Override
public void run() {
sort.Task.run() does not release lock on all exception paths [Troubling(14), Normal confidence]
    try {
        TimeUnit.SECONDS.sleep(1);
        lock.unlock();
```

图 7.5　扫描代码出现问题的警告标识

打开 Bug Explorer 查看简略的警告信息，双击错误定位到编辑代码，如图 7.6 所示。

Problems　Javadoc　Declaration　Debug　Bug Explorer　Properties　Bug Info
JAVA (2)
Troubling (2)
Normal confidence (2)
Invokes run on a thread (did you mean to start it instead?) (1)
sort.Task$Main.main(String[]) explicitly invokes run on a thread (did you mean to start it instead?) [Troubling(14), Normal confidence]
Method does not release lock on all exception paths (1)
sort.Task.run() does not release lock on all exception paths [Troubling(14), Normal confidence]

图 7.6　在 Bug Explorer 中查看代码问题的警告信息

也可右击该警告信息，在快捷菜单中选择 Show Bug Info 命令查看详细的错误信息，如图 7.7 所示。

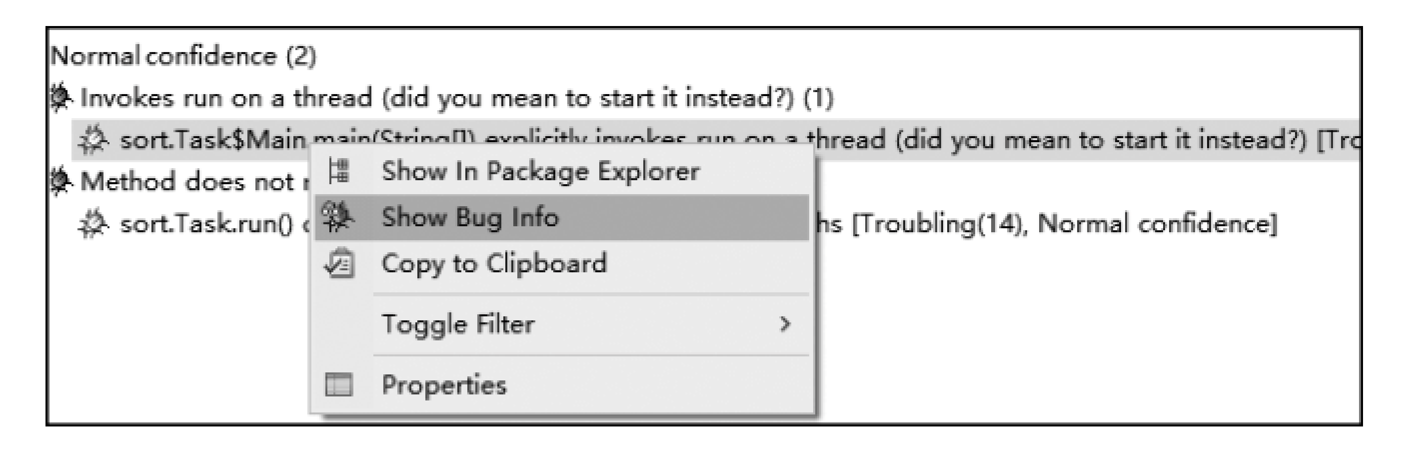


图 7.7　查看详细的错误信息

根据提示，修改代码如下：

```
import java.util.concurrent.TimeUnit;
import java.util.concurrent.locks.ReentrantLock;
//1. 创建一个类,名为 Task,扩展 Runnable 接口
public class Task implements Runnable {
    //2. 声明一个 ReentrantLock 属性,名为 Lock.private
    ReentrantLock lock;//3. 实现类的构造函数。public
    Task(ReentrantLock lock){
```

```
            this.lock=lock;
        }
        //4. 实现 run()方法。获取 lock 的控制,让线程休眠 2 秒再释放 lock
        @Override
        public void run(){
            lock.lock();
            try {
                TimeUnit.SECONDS.sleep(1);
            } catch(InterruptedException e){
                e.printStackTrace();
            }finally {
                lock.unlock();
            }
        }
        //5.创建例子的主类通过创建一个类,名为 Main 并添加 main()方法
        public static class Main {
            public static void main(String[] args){
                //6. 声明并创建一个 ReentrantLock 对象,名为 lock
                ReentrantLock lock=new ReentrantLock();
                //7. 创建 10 个 Task 对象和 10 个线程来执行这些任务。调用 run()方法开始
                //线程
                for(int i=0; i<10; i++){
                    Task task=new Task(lock);
                    Thread thread=new Thread(task);
                    task.run();
                }
            }
        }
    }
```



缺陷管理软件 Bugzilla

实验目的：

(1) 理解 Bugzilla 的功能。

(2) 熟练掌握 Bugzilla 的操作步骤。

实验环境：Bugzilla 软件。

8.1 Bugzilla 简介

Bugzilla 是 Mozilla 公司提供的一个开源的缺陷跟踪工具，拥有大量用户。作为一个产品缺陷的记录及跟踪工具，它能够为软件组织建立一个完善的缺陷跟踪体系，包括报告缺陷，查询缺陷记录并产生报表，处理解决缺陷，管理员系统初始化和设置等。Bugzilla 具有如下特点：

- 基于 Web 方式，运行方便快捷，管理安全。
- 提供强大的查询匹配能力，能根据各种条件组合进行缺陷查询，记忆搜索条件。
- 缺陷从最初的报告到最后的关闭都有详细的操作记录，确保缺陷不会被忽略，并允许用户在检查缺陷状态时获取历史记录。
- 自带基于当前数据库的报表生成功能，主要生成两类图表：基于表格的视图和图形视图(条形图、线图、饼状图)。
- 具有灵活和完善的权限管理功能，管理员可以根据需要定义由个人或者小组构成的访问组。可以定义一组特殊用户，他们所发表的评论和附件只能被组内成员访问。
- 模型化的验证模块，使用户方便地添加所需的系统验证。Bugzilla 已经内置了对 MySQL 和 LDAP 授权验证的支持。
- 自动发送 E-mail 通知相关人员。根据设定的不同责任人，自动发送最新的动态信息，有效地帮助测试人员和开发人员进行沟通。
- 评论回复连接。对缺陷的评论提供直接的页面连接，帮助复查人员评审缺陷。

Bugzilla 的中文网址为 http://www.bugzilla.cn/，如图 8.1 所示。



图 8.1　Bugzilla 的中文网站主页

8.2　Bugzilla 的缺陷处理流程

Bugzilla 的缺陷处理流程包括以下步骤：

(1) 测试人员或开发人员发现缺陷后，判断属于哪个模块的问题，填写缺陷报告后，通过 E-mail 通知项目组长或直接通知开发者。

(2) 项目组长根据具体情况，将缺陷分配给开发者。

(3) 开发者收到 E-mail 信息后，判断是否为自己的修改范围。

- 若不是，重新分配给项目组长或应该负责的开发者。
- 若是，进行处理，解决缺陷并给出解决方法(可创建补丁附件及补充说明)。

(4) 测试人员查询开发者已修改的缺陷，进行测试(可创建 test case 附件)。

- 经验证无误后，修改状态为 Verified。待整个产品发布后，修改为 Closed。
- 若还有问题，重新打开缺陷，状态重新变为 New，并发邮件通知相关人员。

(5) 如果这个缺陷一周内一直没被处理过，Bugzilla 就会一直用 E-mail 通知它的属主，直到缺陷被处理。

8.3　环 境 搭 建

Bugzilla 可以在 Linux 平台和 Windows 平台上使用，本书重点介绍 Bugzilla 在 Windows 平台的使用情况。Bugzilla 的安装需要如下软件：MySQL 数据库、ActivePerl 软件、Bugzilla 安装包、IIS 组件，整个安装过程需要联网。下面介绍具体安装步骤。

8.3.1　MySQL 数据库

MySQL 下载网址为 http://dev.mysql.com/downloads/installer/，安装到 C:\MySQL

目录下，如图 8.2 所示。

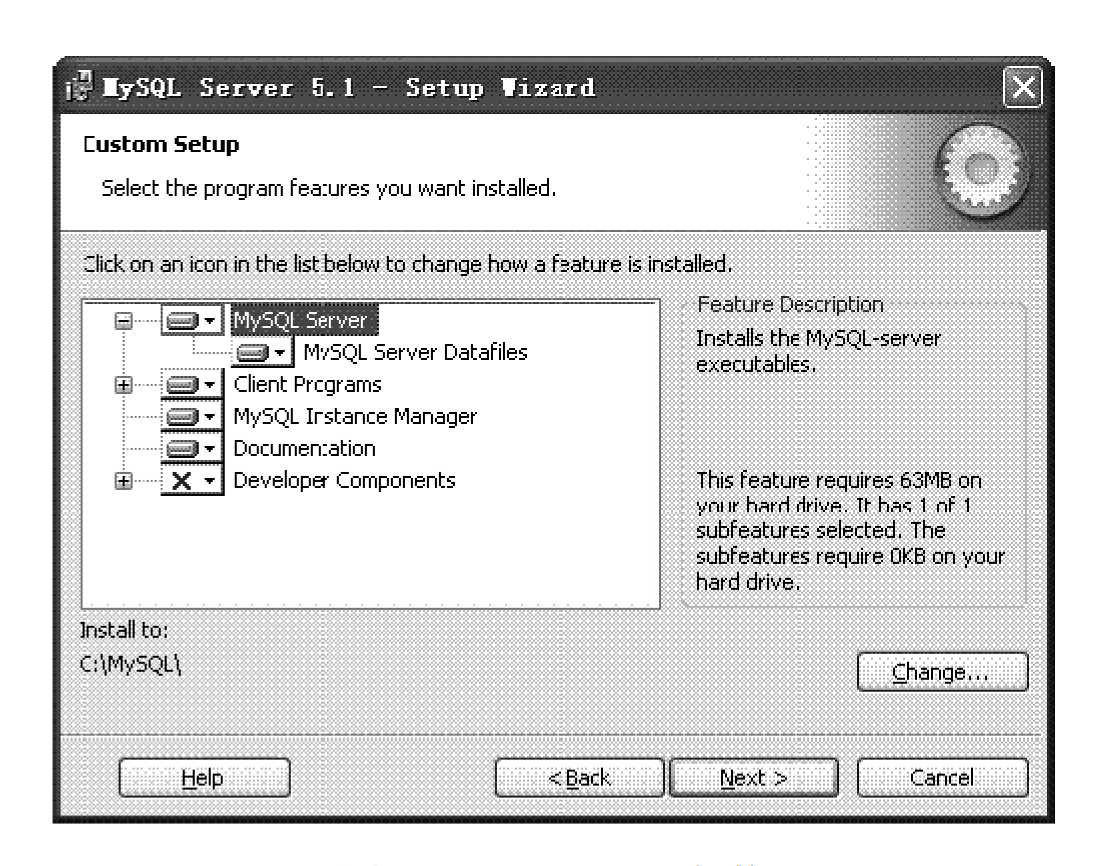


图 8.2 MySQL 安装

安装好后，在 MySQL 服务器中创建数据库 bugs 和用户 bugs，并为该用户授予相应的权限，如图 8.3 所示。

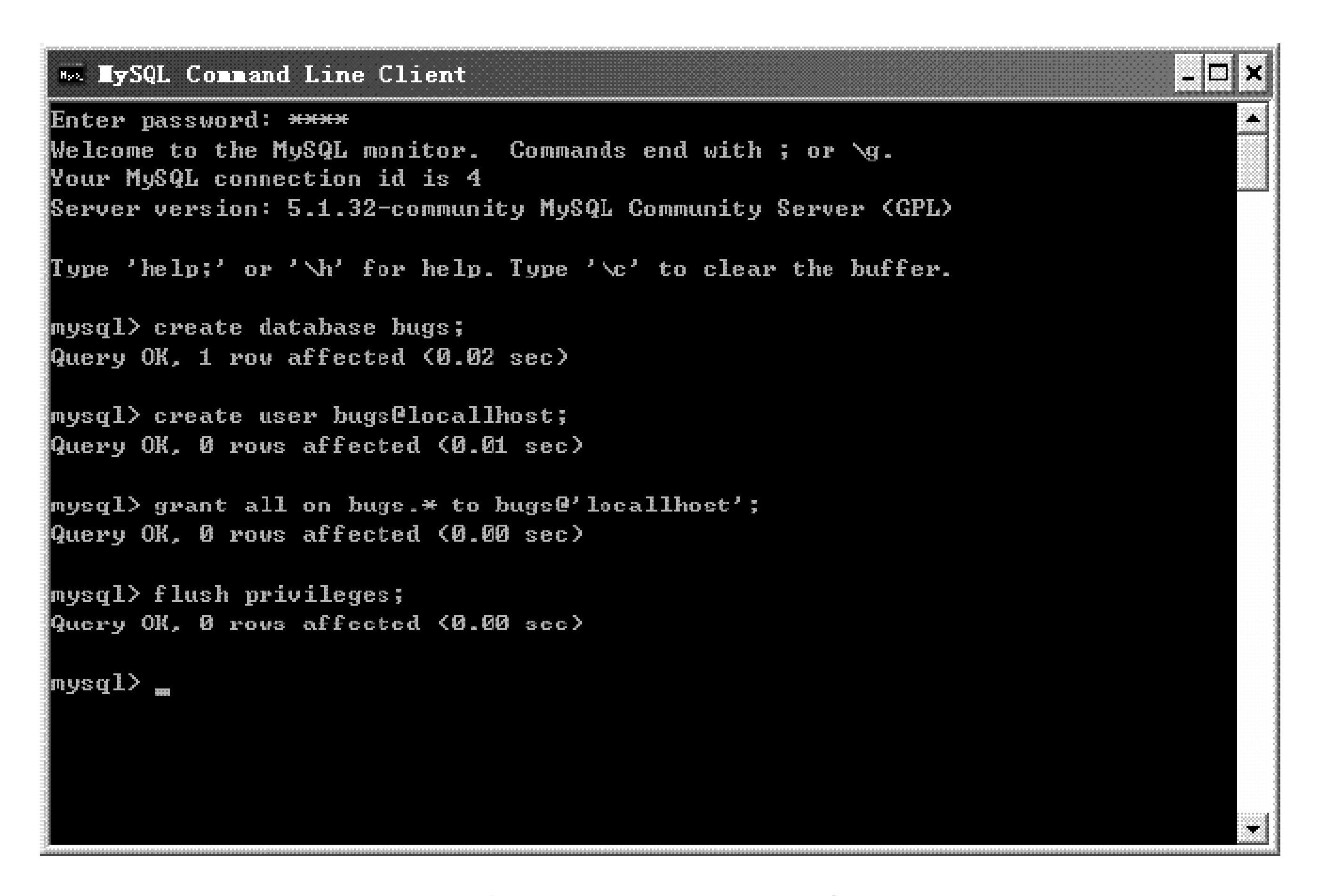


图 8.3 MySQL 运行

命令说明如下：

create database bugs;	创建一个数据库 bugs
create user bugs@ localhost;	创建一个用户 bugs
grant all on bugs. * to bugs@ 'localhost';	为用户 bugs 授权
flush privileges;	刷新用户权限

8.3.2 ActivePerl

下载 ActivelPerl-5.. 6. 1. 633-MSWin32-x86. msi 或以上版本，安装 ActivePerl，如图 8.4 所示。

不用修改默认选项，继续进入下一步，直到安装结束。在命令行输入 perl -v 检测



图 8.4　ActivePerl 安装

Perl 是否安装成功，如图 8.5 所示。

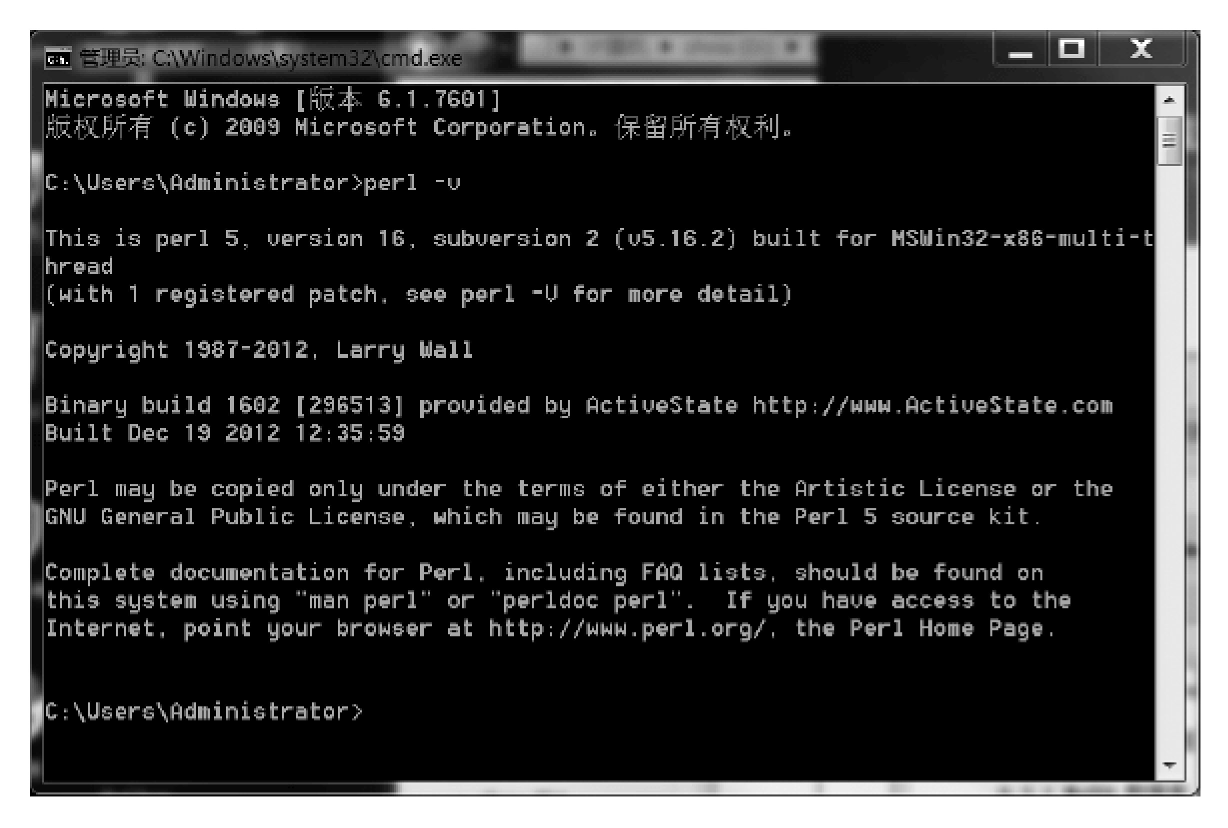


图 8.5　Perl 安装成功

8.3.3　Bugzilla 安装包

下载 Bugzilla 的网址是 https://www.bugzilla.org/，下载 Bugzilla 4.2 或者更高版本，将 Bugzilla 安装包解压并复制到 C 盘根目录下。使用 Bugzilla 自带的 checksetup.pl 来安装 Bugzilla 所需的 Perl 模块，如图 8.6 所示。

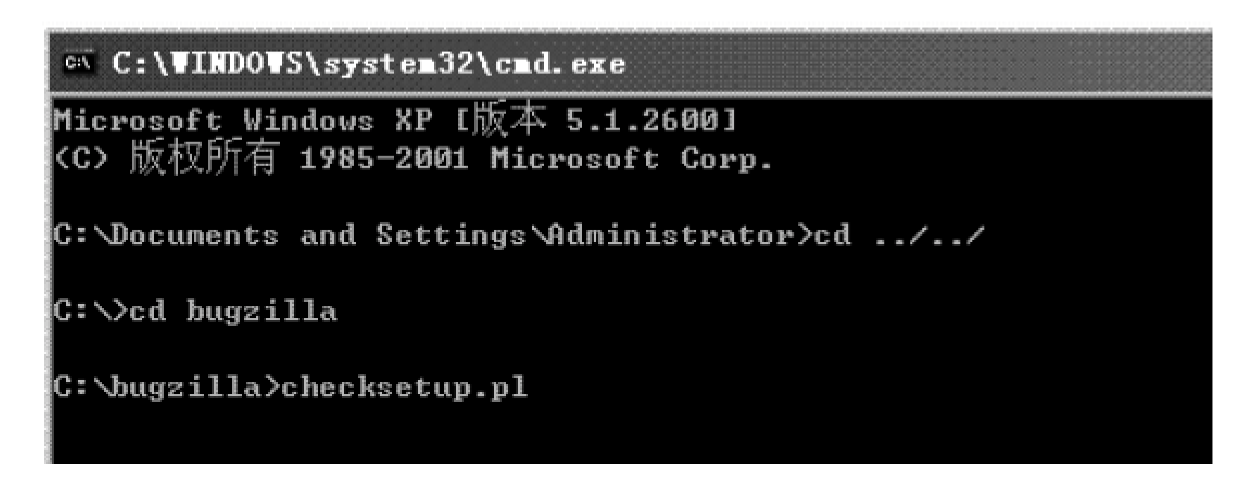


图 8.6　checksetup.pl 命令

安装 Template-CD 模块，在命令行输入 ppm install Template-CD 命令，完成该模块的安装，如图 8.7 所示。

```
C:\WINDOWS\system32\cmd.exe - checksetup.pl
*************************************************************************
*      Template-GD * Graphical Reports                                  *
*        SOAP-Lite * XML-RPC Interface                                  *
*      TheSchwartz * Mail Queueing                                      *
*   Daemon-Generic * Mail Queueing                                      *
*         mod_perl * mod_perl                                           *
* Apache-SizeLimit * mod_perl                                           *
*************************************************************************
* Note For Windows Users                                                *
*************************************************************************
* In order to install the modules listed below, you first have to run  *
* the following command as an Administrator:                            *
*                                                                       *
*   ppm repo add theory58S http://cpan.uwinnipeg.ca/PPMPackages/10xx/
*************************************************************************
COMMANDS TO INSTALL OPTIONAL MODULES:

    Template-GD: ppm install Template-GD
      SOAP-Lite: ppm install SOAP-Lite
    TheSchwartz: ppm install TheSchwartz
 Daemon-Generic: ppm install Daemon-Generic
       mod_perl: ppm install mod_perl
Apache-SizeLimit: ppm install Apache-SizeLimit
```

图 8.7　安装 Template-CD 模块

安装成功后，会在 bugzilla 目录下生成 localconfig 文件，打开 localconfig 文件，将其中的 $db_port = 0;改为 $db_port = 3306;，将 $index_html = 0;改为 $index_html = 1;，在命令行下再次运行 checksetup.pl 将会生成和数据库有关的数据表，然后安装程序会要求填入服务器主机地址，如图 8.8 所示。

```
C:\WINDOWS\system32\cmd.exe - checksetup.pl
Creating ./data/extensions/additional...
Creating ./data/mailer.testfile...
Creating index.html...
Creating ./Bugzilla/.htaccess...
Creating ./data/.htaccess...
Creating ./data/attachments/.htaccess...
Creating ./data/webdot/.htaccess...
Creating ./graphs/.htaccess...
Creating ./lib/.htaccess...
Creating ./template/.htaccess...
Creating contrib/.htaccess...
Creating t/.htaccess...
Creating xt/.htaccess...

Bugzilla requires an SMTP server to function on Windows.
Please enter your SMTP server's hostname: 192.168.[redacted]
Precompiling templates...done.
Initializing "Dependency Tree Changes" email_setting ...
Deriving regex group memberships...

Looks like we don't have an administrator set up yet. Either this is
your first time using Bugzilla, or your administrator's privileges
might have accidentally been deleted.

Enter the e-mail address of the administrator: _
```

图 8.8　配置 localconfig 文件

在网址栏上输入 http://192.168.1.1/bugzilla，登录 Bugzilla，如图 8.9 所示。

8.3.4　IIS

Windows 10 默认不安装 IIS，需要单独安装，方法为：在“控制面板”中选择“程序”→“打开或关闭 Windows 功能”，选择安装 IIS 的相应功能，如图 8.10 所示。对于不同版本



图 8.9　登录 Bugzilla

的 Windows，IIS 设置有些不同，依照具体情况设置。

进入 IIS 进行配置，步骤如下。

步骤 1：在“运行”对话框中输入 inetmgr 命令，如图 8.11 所示。



图 8.10　选择要安装的 IIS 功能

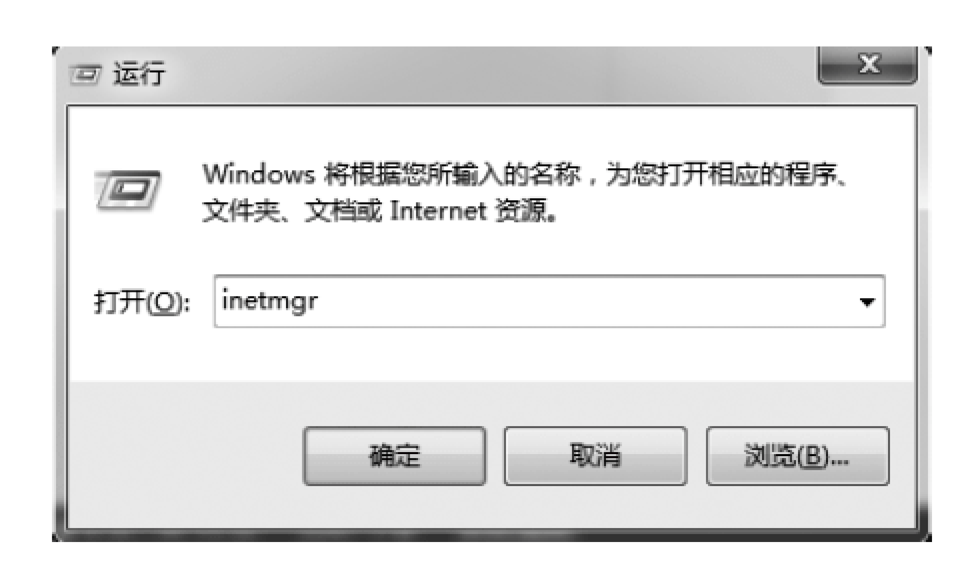


图 8.11　运行 inetmgr 命令

步骤 2：在 Default Web Site 上右击，在快捷菜单中选择“添加应用程序”命令，如图 8.12 所示。

步骤 3：单击 bugzilla，如图 8.13 所示。

步骤 4：双击“默认文档”，在右上角单击“添加”，打开“添加默认文档”对话框，如图 8.14 所示。

步骤 5：单击 Default Web Site，在功能视图中找到“处理程序映射”，在右上角选择“添加脚本映射”，打开“添加脚本映射”对话框，如图 8.15 所示。



图 8.12 添加应用程序



图 8.13 IIS 配置



图 8.14 添加默认文档



图 8.15 添加脚本映射

步骤 6：在 Default Web Site 中再次单击“添加脚本映射”，打开“添加脚本映射”对话框，如图 8.16 所示。

步骤 7：单击 bugzilla，在功能视图中找到“处理程序映射”，在右上选择“添加脚本映射”，打开“添加脚本映射”对话框，如图 8.17 所示。



图 8.16　再次添加脚本映射



图 8.17　第三次添加脚本映射

步骤 8：在“运行”对话框中输入 iisreset 命令，重启 IIS，如图 8.18 所示。

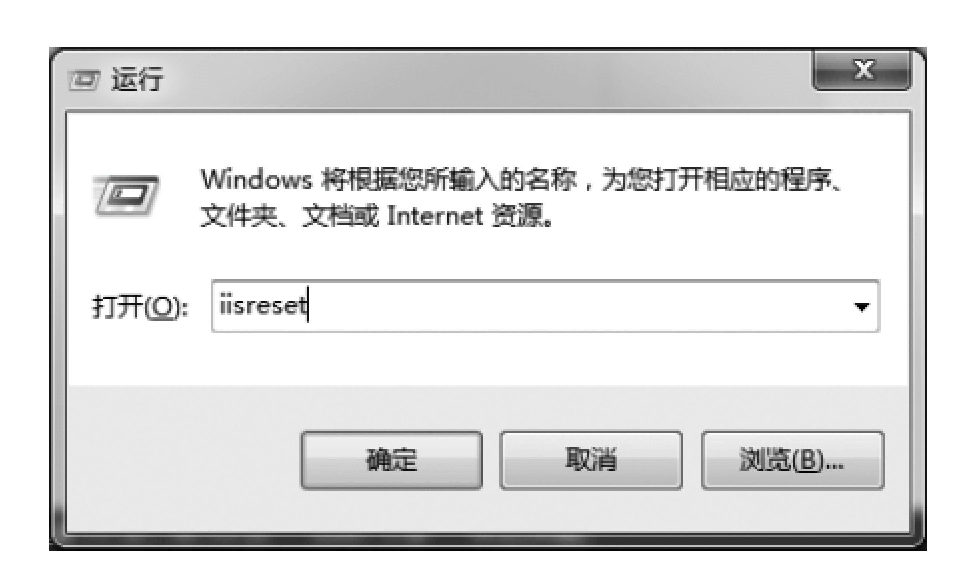


图 8.18　重启 IIS

步骤 9：访问 Bugzilla，输入网址 http://127.0.0.1/bugzilla/或 http://127.0.0.1/bugzilla/index.cgi。

8.4　实验内容

使用 Bugzilla 管理缺陷的操作步骤如下。

1. 新建一个用户账号

单击 Open a new Bugzilla account 链接，输入用户的 E-mail 地址，单击 Create Account 按钮。稍后，用户会收到一封电子邮件，邮件中包含用户的登录账号(与 E-mail 相同)和口令，这个口令是 Bugzilla 系统随机生成的，可以根据需要进行变更。

2. 报告缺陷

在 Bugzilla 界面中输入缺陷的相关信息，如图 8.19 所示。

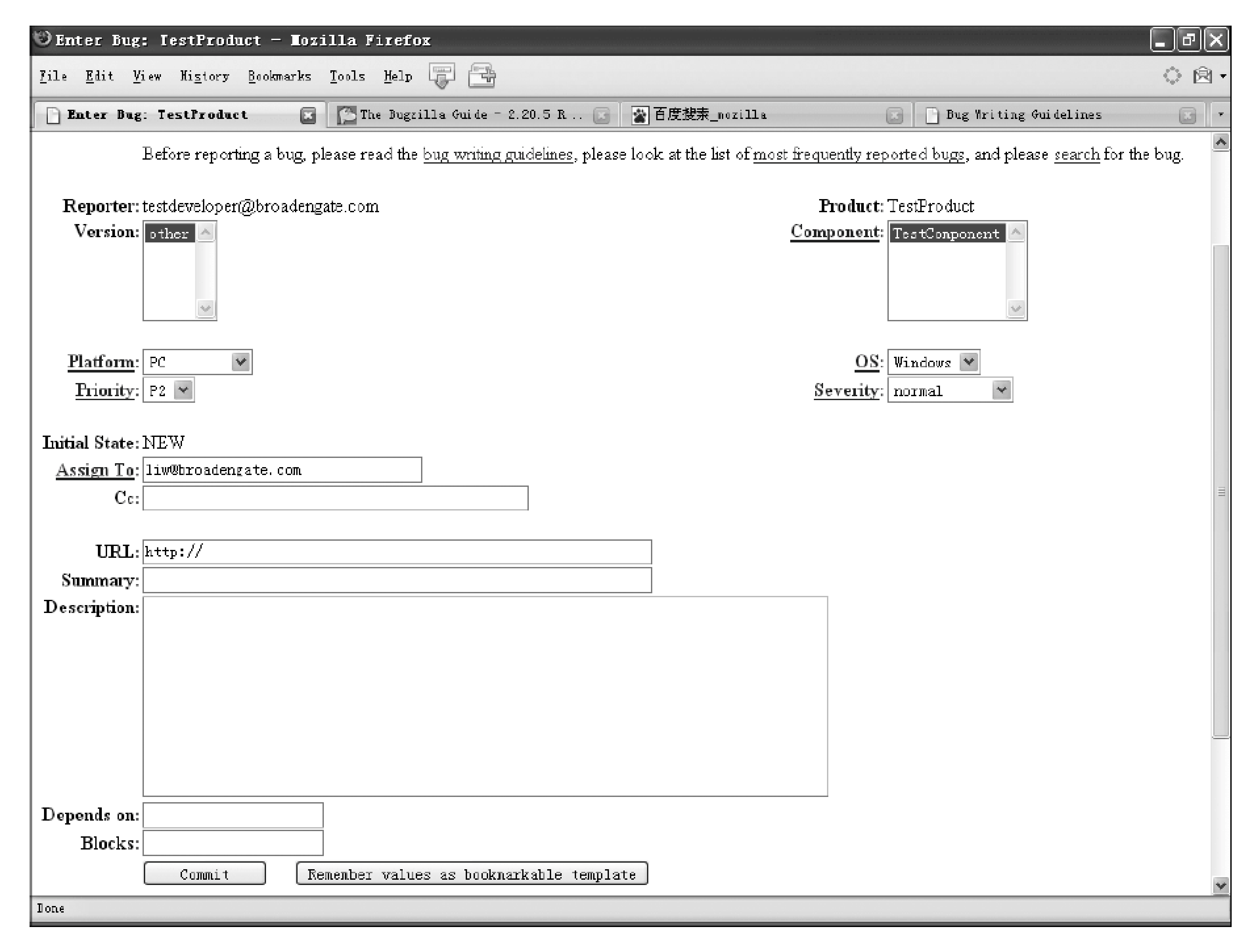


图 8.19　在 Bugzilla 中报告缺陷的页面

相关参数如下：

(1) Priority 是缺陷的优先级。

(2) Initial State 是缺陷的初始状态，有以下几种状态：

- Unconfirmed：待确认的。
- New：新提交的。
- Assigned：已分配的，指已分配给相关人员进行修复。
- Reopened：因问题未解决而重新打开的。
- Resolved：缺陷已修复而待返测的。
- Verified：已经过返测而待归档的。
- Closed：已归档(关闭)的。

(3) Assign To 是将缺陷分配给哪一个人员进行处理。

(4) Cc 指可同时将缺陷报告抄送给哪些人员，如果为多人，需用逗号隔开。

(5) URL 是一个超链接地址，引导处理人找到与缺陷报告相关联的信息。

(6) Summary 是概述部分，保证处理人能够理解提交者发现的问题。

(7) Description 是缺陷的详细描述，一般要说明下列情况：

- 发现问题的步骤。

- 执行上述步骤后出现的情况。
- 期望出现的正确结果。

(8) Depends on 是指缺陷的依赖关系，如果这个缺陷必须在其他缺陷修改以后才能够修改，则在这里填写缺陷的编号。

(9) Blocks 是指如果该缺陷影响其他缺陷的修改，则在这里填写被影响的缺陷编号。

(10) Component 是指缺陷所在的软件模块。

(11) Severity 指缺陷的严重程度，可选择以下选项：

- Blocker，阻碍开发和/或测试工作。
- Critical，关键性缺陷，如死机、丢失数据、内存溢出等。
- Major，较大的功能缺陷。
- Normal，普通的功能缺陷。
- Minor，较小的功能缺陷。
- Trivial，产品外观上的问题或不影响使用的毛病，如菜单的文字拼写问题等。

填写以上信息后，单击 Commit 按钮提交缺陷，Bugzilla 会自动发送邮件通知相关人员。

3. 确认缺陷是否存在

查询状态为 Unconfirmed 的缺陷，对其进行确认操作，具体操作方法是：选中 Confirm bug(change status to New)后，单击 Commit 按钮，操作结果为缺陷的状态变为 New。

4. 解决缺陷

登录 Bugzilla 系统后，单击浏览器下方的 Saved Searches：My Bugs 按钮进入缺陷管理界面，选择缺陷进行修复，给出解决方式并填写 Additional Comments，如图 8.20 所示。

缺陷的解决方式有以下几种：

- FIXED：问题已经修复。
- INVALID：描述的问题不是一个缺陷。
- WONTFIX：描述的问题不需要修复。
- LATER：描述的问题将不会在产品的这个版本中解决。
- DUPLICATE：描述的问题与以前的某个缺陷重复。
- WORKSFORME：无法重现缺陷。

5. 验证已修复的缺陷

开发人员处理完缺陷并提交之后，测试人员查询开发者已修改的缺陷，即 Status 为 Resolved，Resolution 为 Fixed 的缺陷，进行回归测试。经验证无误后，将缺陷的状态改为 Verified。若还有问题，则重新打开缺陷，状态重新变为 New，并发邮件通知开发人员，如图 8.21 所示。

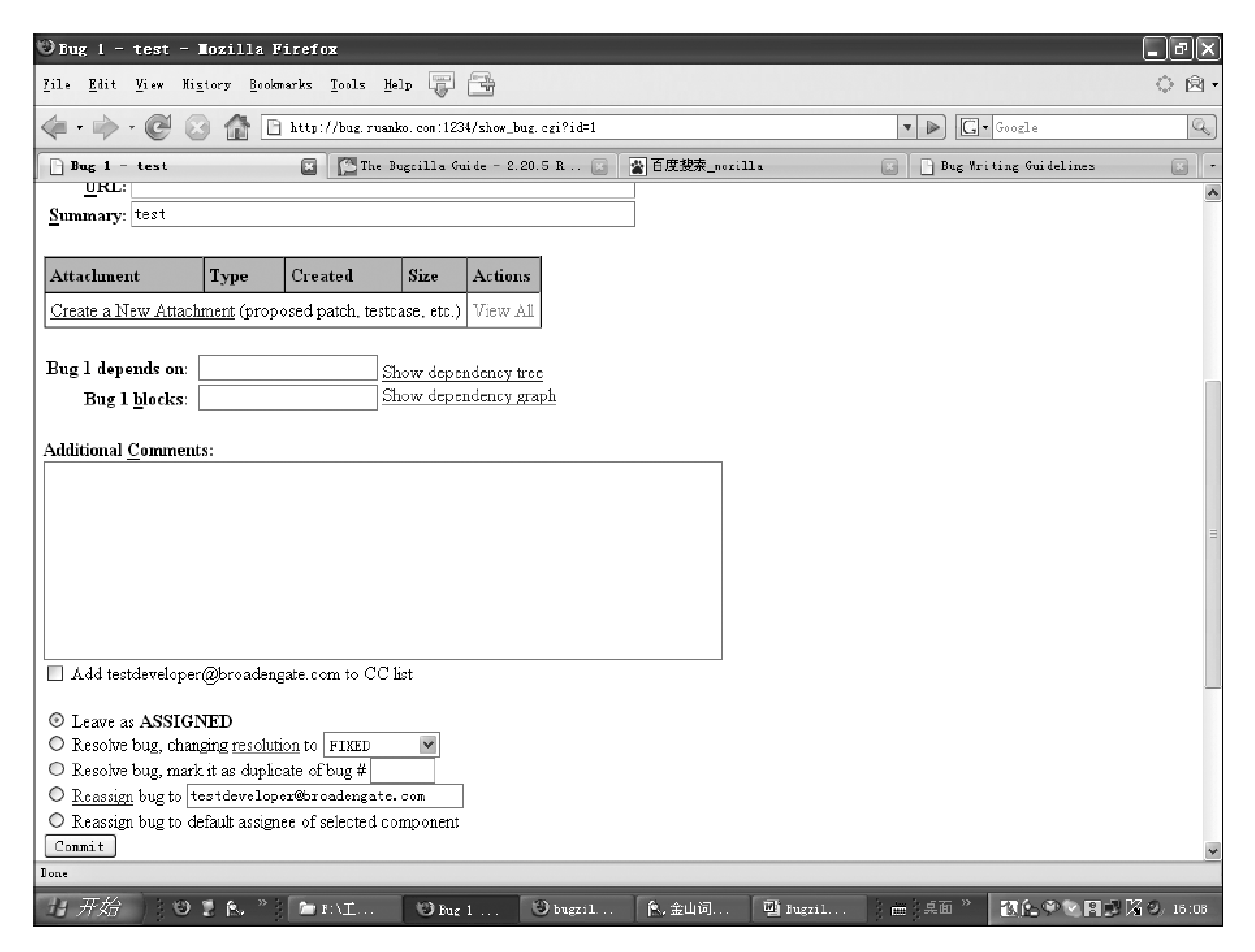


图 8.20　处理缺陷

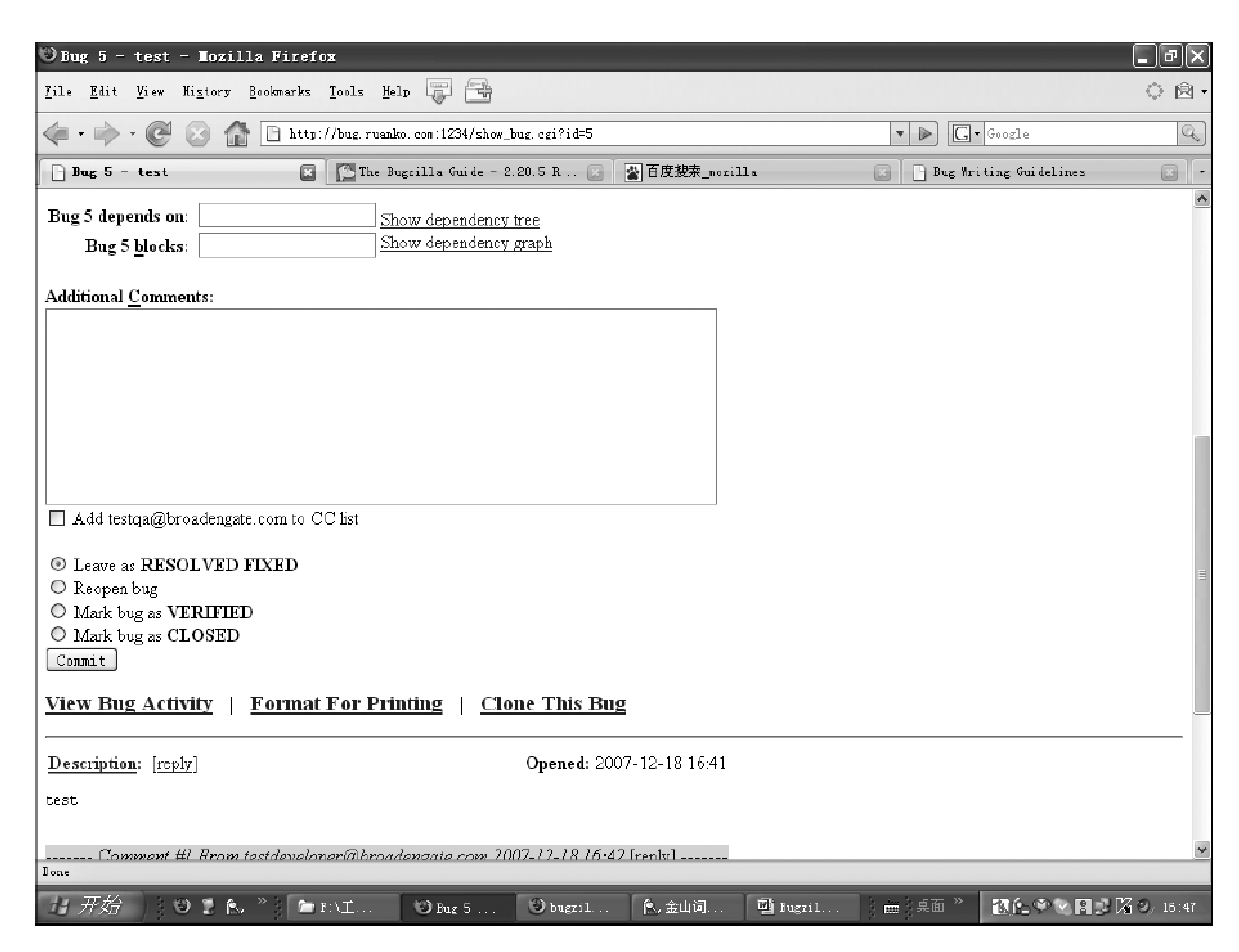


图 8.21　验证已修改的缺陷

6. 查找缺陷

使用 Bugzilla 的查询页面查找到系统中所有的缺陷信息。缺陷报告中所有的字段信息都可作为查询条件，对于某些字段，可以选择多个值，在这种情况下，Bugzilla 会返回与任一值匹配的缺陷记录。

将一个已执行的查询保存下来，成为一个“已保存查询”(Saved Search)，显示在查询页面的页脚处，供以后重复使用。使用布尔表达式将多个查询条件组合在一起，构成高级查询。

7. 生成报表

当使用查询功能得到缺陷数据集后，可在此缺陷数据集的基础上生成报表。Bugzilla 的报表分为两类：基于 HTML 表格的报表和基于图形的报表。其中基于图形的报表可绘制线图、饼状图和柱状图。例如，当使用查询功能查询出某一产品中的所有缺陷记录后，可使用报表显示出各构件的缺陷分布状况，从而发现哪些模块的质量存在严重问题。当定义好报表参数后，单击 Generate Report 生成报表，报表的形式可在 HTML 表格、线图、饼图和柱状图之间切换。

实验9

移动测试软件 Appium

实验目的：

（1）了解并熟悉移动应用测试工具 Appium。

（2）掌握利用 Appium 进行 Android 或 iOS 应用测试。

（3）测试全国大学生软件测试大赛的大角虫软件，测试其登录功能。

实验环境：Appium 软件。

9.1 实验内容

实验内容如下：

（1）创建测试项目。

（2）针对待测软件编写测试脚本。

（3）掌握常见的真机测试技巧。

9.2 环境搭建

9.2.1 JDK 和 Eclipse 安装与配置

去官网下载 JDK 1.7，进行安装，修改 classpath。

在官网下载 Eclipse，版本是 Luna Service Release 2（4.4.2），进行安装。

9.2.2 SDK 安装与配置

在 Android 中文官网下载 SDK，参照 http://blog.csdn.net/slow_liao/article/details/44358971，如图 9.1 所示。

下载之后，配置 ANDROID_HOME 环境变量，并确保 $ANDROID_HOME/platform-tools 和 $ANDROID_HOME/tools 包含在 PATH 中。在命令提示符窗口输入命令：android -h，如果出现如图 9.2 所示的信息，说明配置成功。

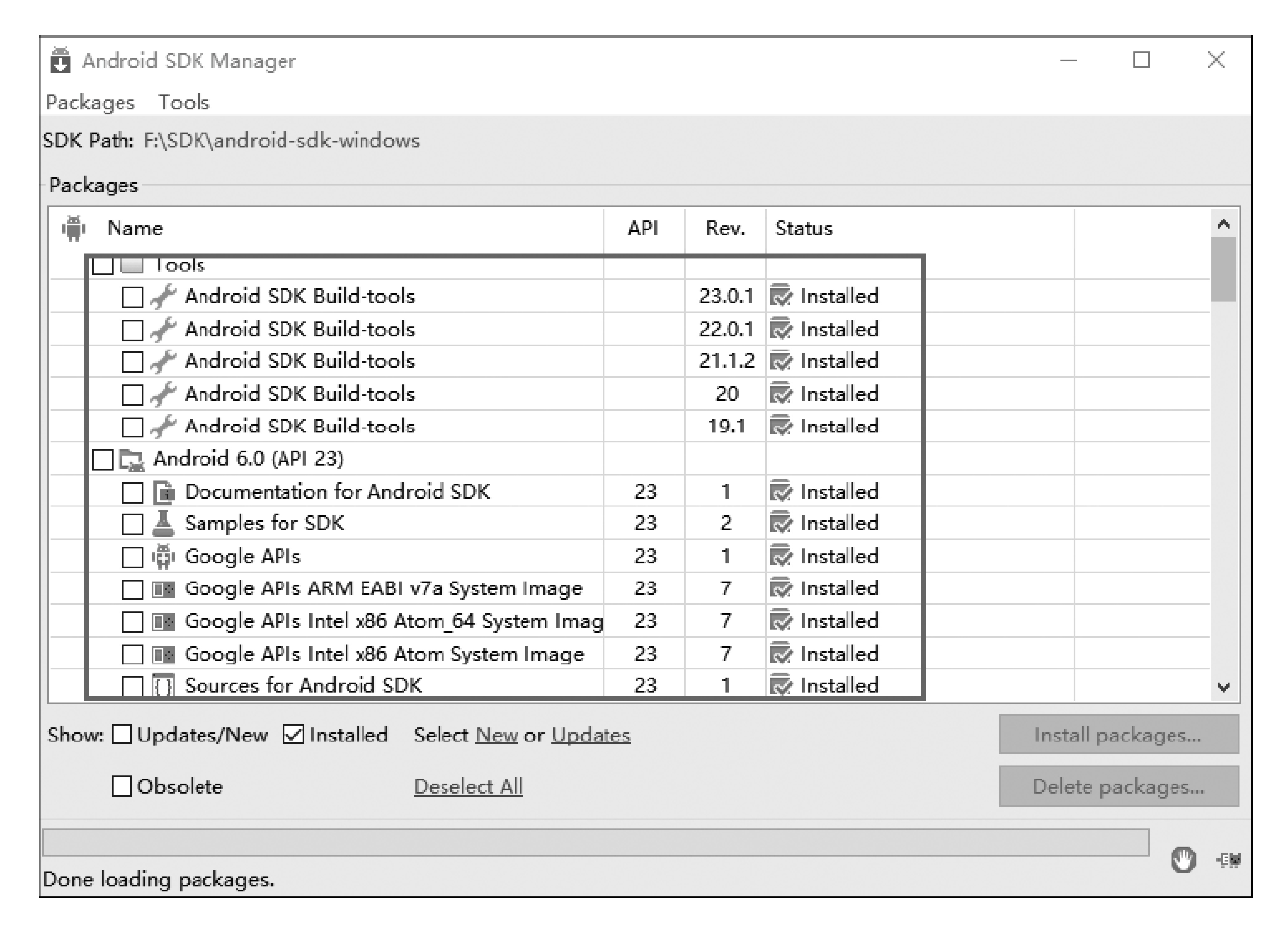


图 9.1　从 Android 中文官网下载的 SDK

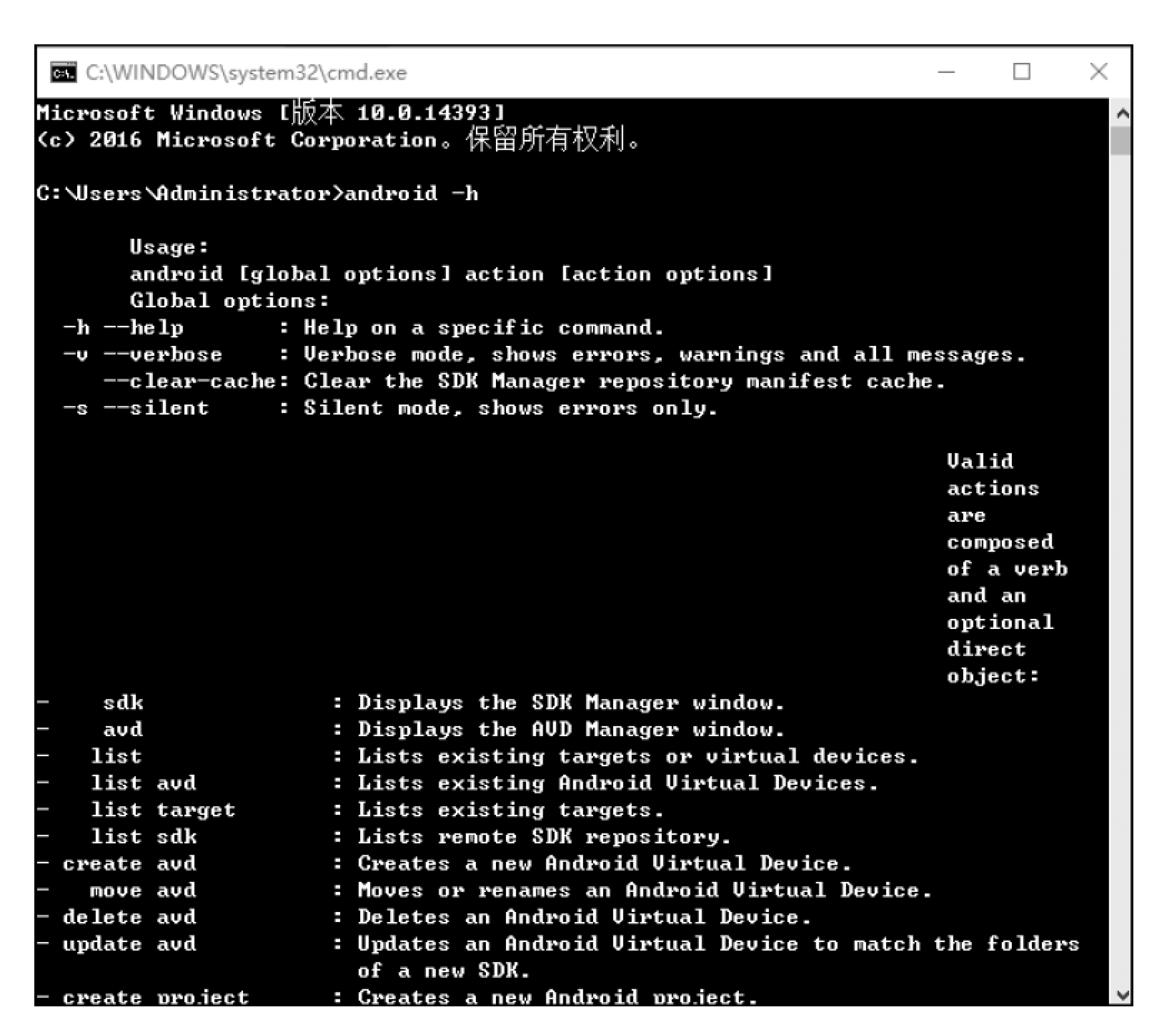


图 9.2　ANDROID_HOME 环境变量配置成功

9.2.3　Appium 的安装与配置

在 Appium 官网 http://appium.io/下载 Appium，其安装到 D:\Appium_1.4\node_modules\.bin 目录，在环境变量 path 中配置 Appium 的安装路径。在 cmd 中输入命令

appium，如果出现如图 9.3 所示的信息，说明配置成功。

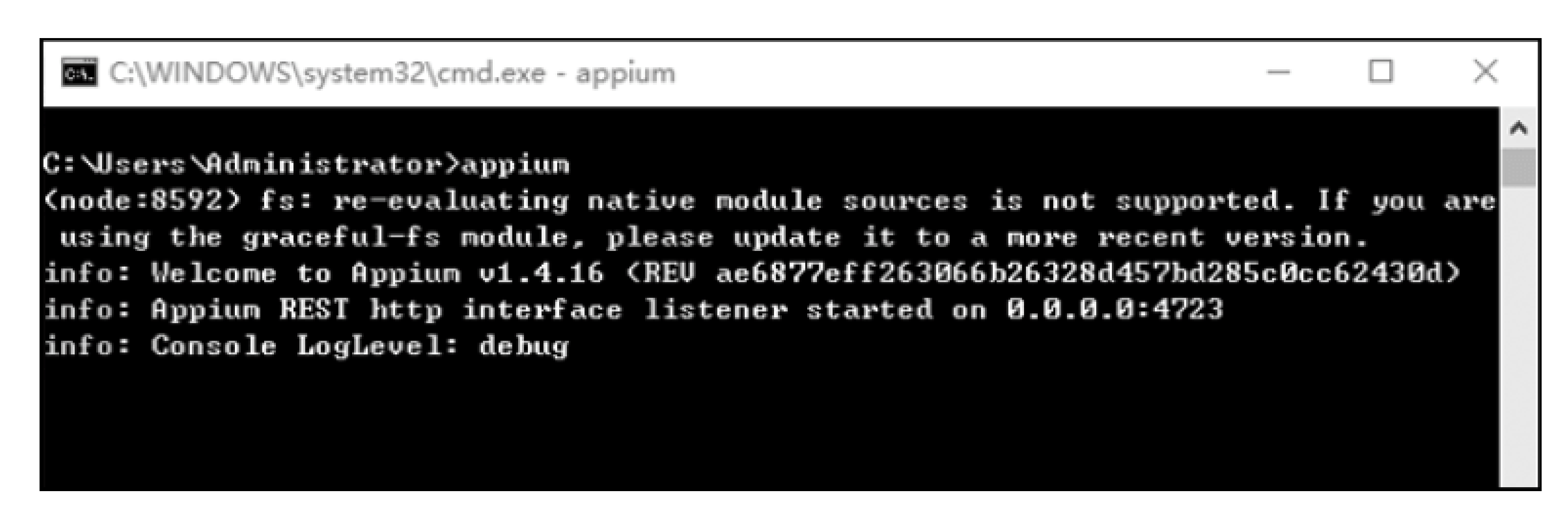


图 9.3　Appium 环境变量配置成功

9.2.4　相关文件和 jar 包下载

（1）下载 selenium-server-standalone-2.51.0.jar 文件。

在网址 http://selenium-release.storage.googleapis.com/index.html 下载 selenium-server-standalone-2.51.0.jar，如图 9.4 所示。

Index of /

Name	Last modified	Size	ETag
2.39	-	-	-
2.40	-	-	-
2.41	-	-	-
2.42	-	-	-
2.43	-	-	-
2.44	-	-	-
2.45	-	-	-
2.46	-	-	-
2.47	-	-	-
2.48	-	-	-
2.49	-	-	-
2.50	-	-	-
2.51	-	-	-
2.52	-	-	-
2.53	-	-	-
3.0-beta1	-	-	-
3.0-beta2	-	-	-
3.0-beta3	-	-	-
3.0-beta4	-	-	-
3.0	-	-	-
icons	-	-	-
index.html	2014-01-13 22:12:39	0.01MB	704b0f841aad1b1428481b7ff3c759c0

图 9.4　selenium-server-standalone-2.51.0.jar 文件

（2）下载 java-client-4.1.2.jar 文件。

输入网址 http://maven.outofmemory.cn/io.appium/java-client/4.1.2/，下载 java-client-4.1.2.jar，如图 9.5 所示。

（3）下载 selenium-2.51.0 文件。

输入网址 http://selenium-release.storage.googleapis.com/index.html? path=2.51/，下载 selenium-2.51.0，解压后 libs 里面的 jar 包如图 9.6 所示。

（4）dx.jar、shrinkedAndroid.jar、apkUtil.jar 在官网下载。

（5）相关文件下载安装，包括 aapt、aapt.exe、libbcc.so、libbcinfo.so、libc++.so、



图 9.5　java-client-4.1.2.jar 文件

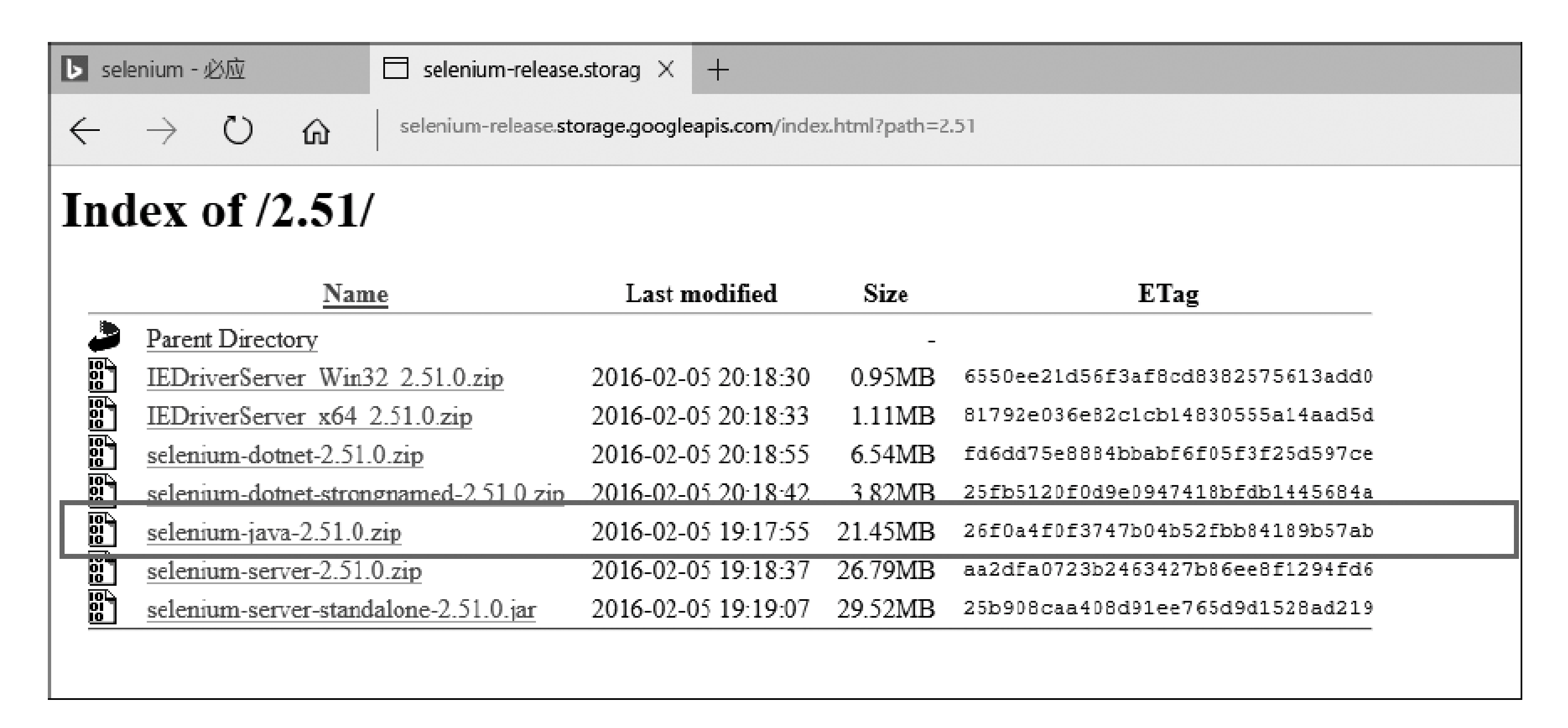


图 9.6　selenium-2.51.0 文件

libclang.so、libLLVM.so。

9.3　实验步骤

9.3.1　测试项目的创建

步骤 1：打开 Eclipse，新建一个 Java 项目，命名为 Test。

步骤 2：右击 Test 项目，在快捷菜单中选择 build path 命令，选择 Add Library 对话框中的 User Library，如图 9.7 所示。

步骤 3：单击 Next 按钮，选择 User Libraries，单击 New 按钮创建 3 个库，如图 9.8 所示。

- Client：创建该库后，单击 Add External JARs 按钮，找到之前下载的 java-client-

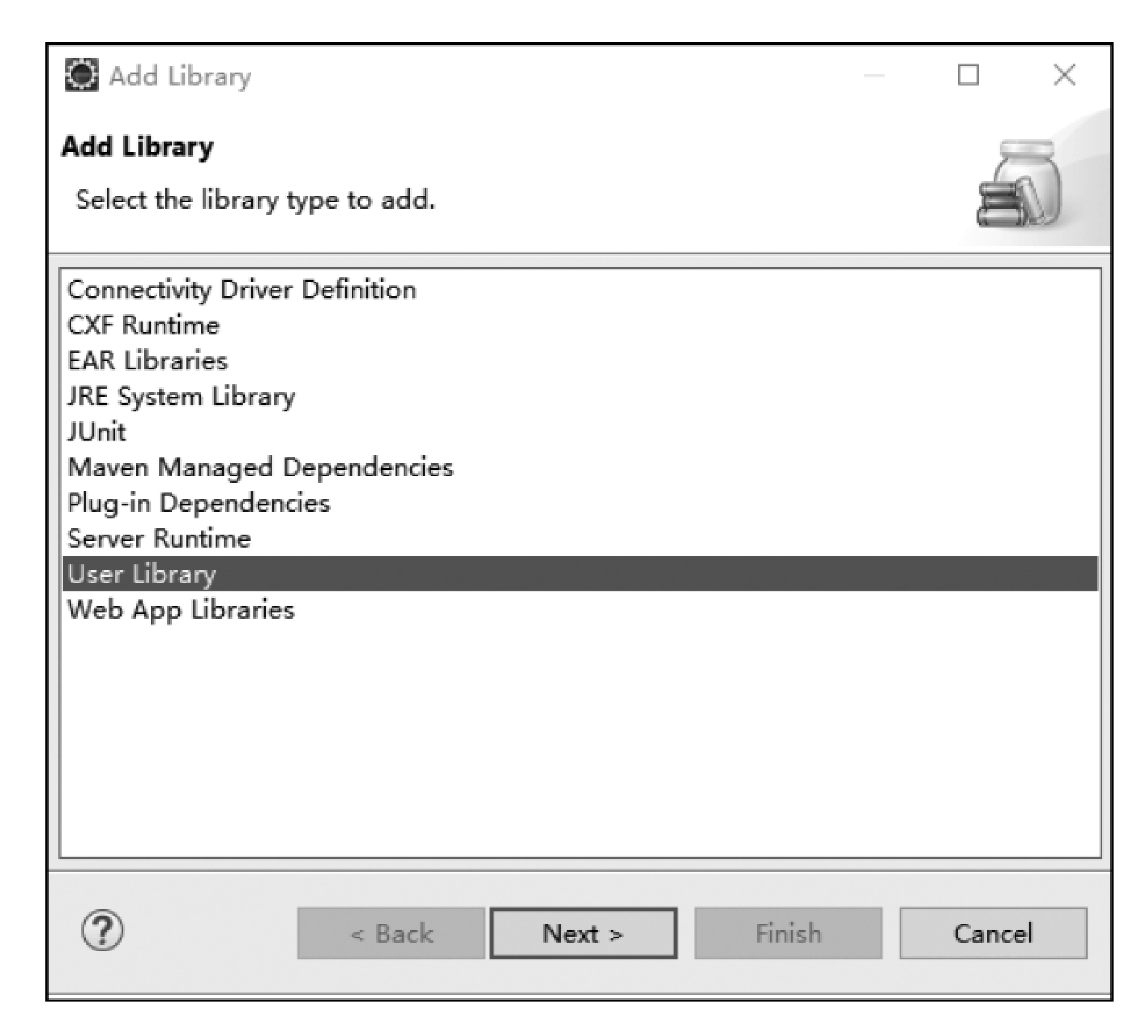


图 9.7　选择 Add Library 对话框中的 User Library

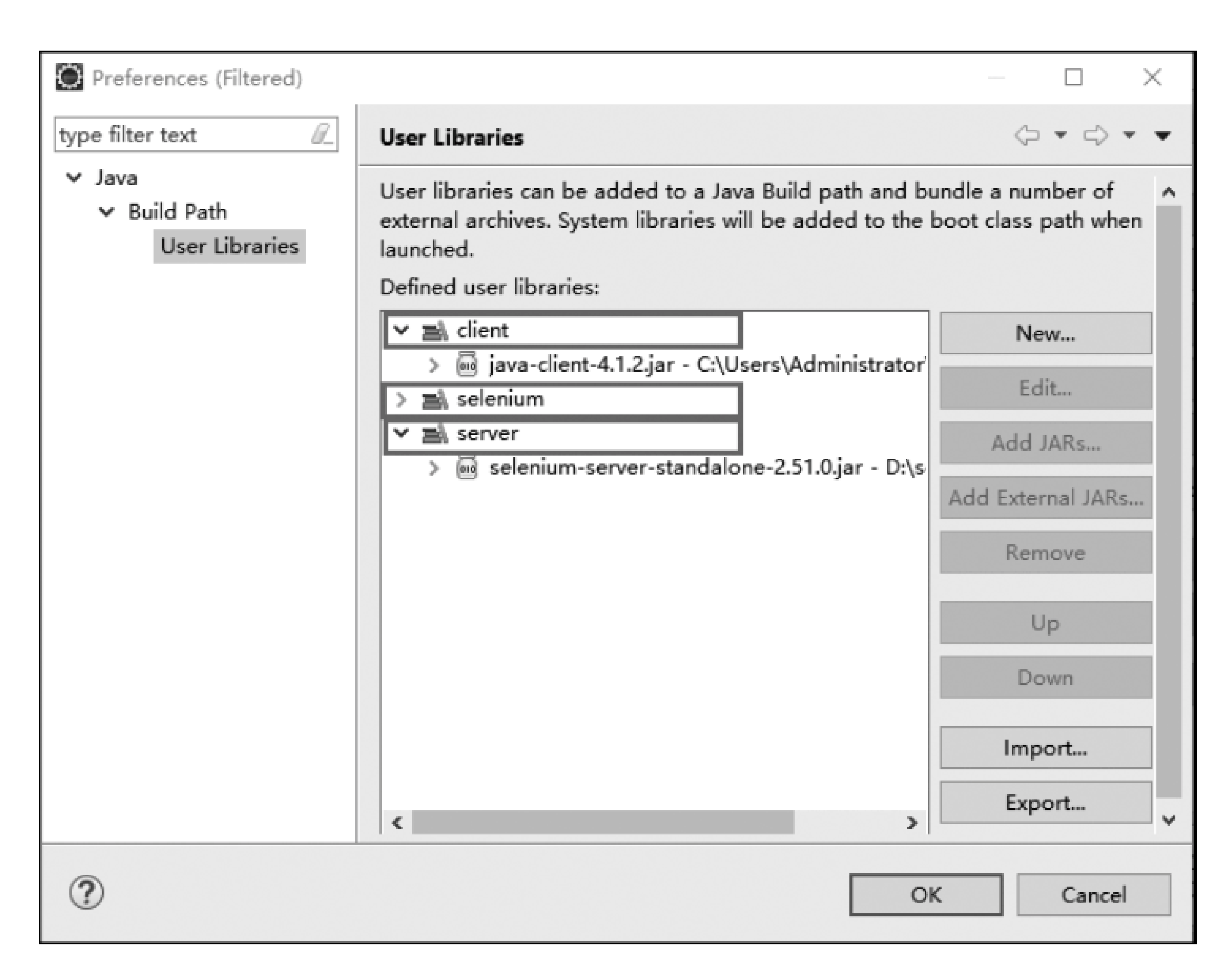


图 9.8　创建 3 个库

4.1.2.jar。

- Server：导入的是之前下载的 selenium-server-standalone-2.51.0.jar。
- Selenium：导入的是之前下载的 selenium-2.51.0 安装包 libs 下的所有 jar 包，最好把 libs 文件夹外面的 jar 包也都导入进去，导入方式同上。

步骤 4：将创建的 3 个库全部选中导入项目。

步骤 5：将之前下载的 dx.jar、shrinkedAndroid.jar、apkUtil.jar 都通过 Build Path 中的 Add External JARs 导入到项目里面。

步骤 6：在 Text 项目下新建一个名为 lib 的文件夹，直接将 aapt 和 aapt.exe 复制到 lib 文件夹下。

步骤 7：在 lib 文件夹下新建名为 lib 的包，将之前下载的所有.so 文件复制到该包下。至此，项目结构如图 9.9 所示。

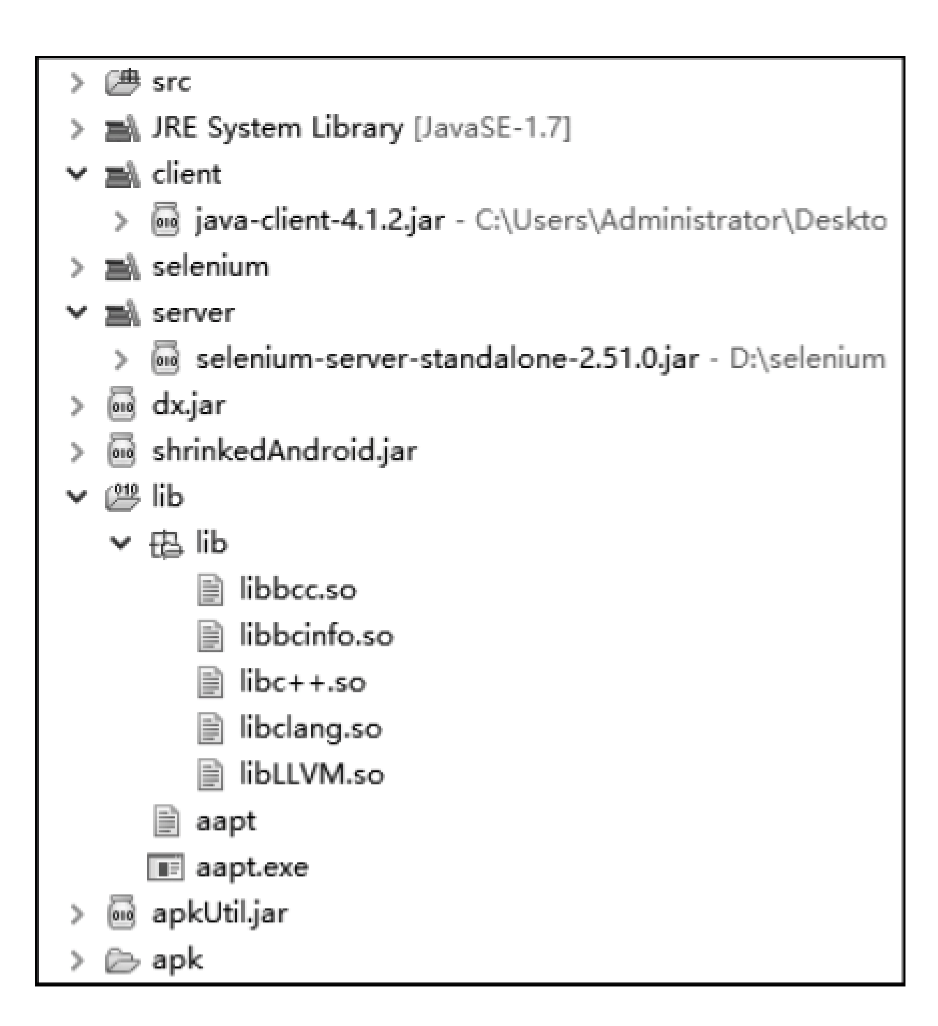


图 9.9 项目结构

9.3.2 针对待测软件编写测试脚本

步骤 1：本书配套电子资源提供了 2016 年全国大学生软件测试大赛使用的大角虫软件安装包，即 apk 文件(该 apk 是一个漫画阅读类软件，仅做测试用)，取名为 dajiaochong.apk。



图 9.10 大角虫软件的安装包

步骤 2：在 Test 项目下新建名为 apk 的文件夹，直接将 dajiaochong.apk 文件复制过来。

步骤 3：在 src 文件夹下创建名为 com.mooctest 的包，Java 文件命名为 Main.java。项目结构如图 9.11 所示。

步骤 4：将 Android 手机连接到计算机上，开启“开发者模式”，打开 USB 调试功能，如图 9.12 所示。在命令提示符窗口中输入命令 adb devices，如果 SDK 安装配置成功，则会出现该手机的 udid 号，如图 9.13 所示，表示手机连接成功。

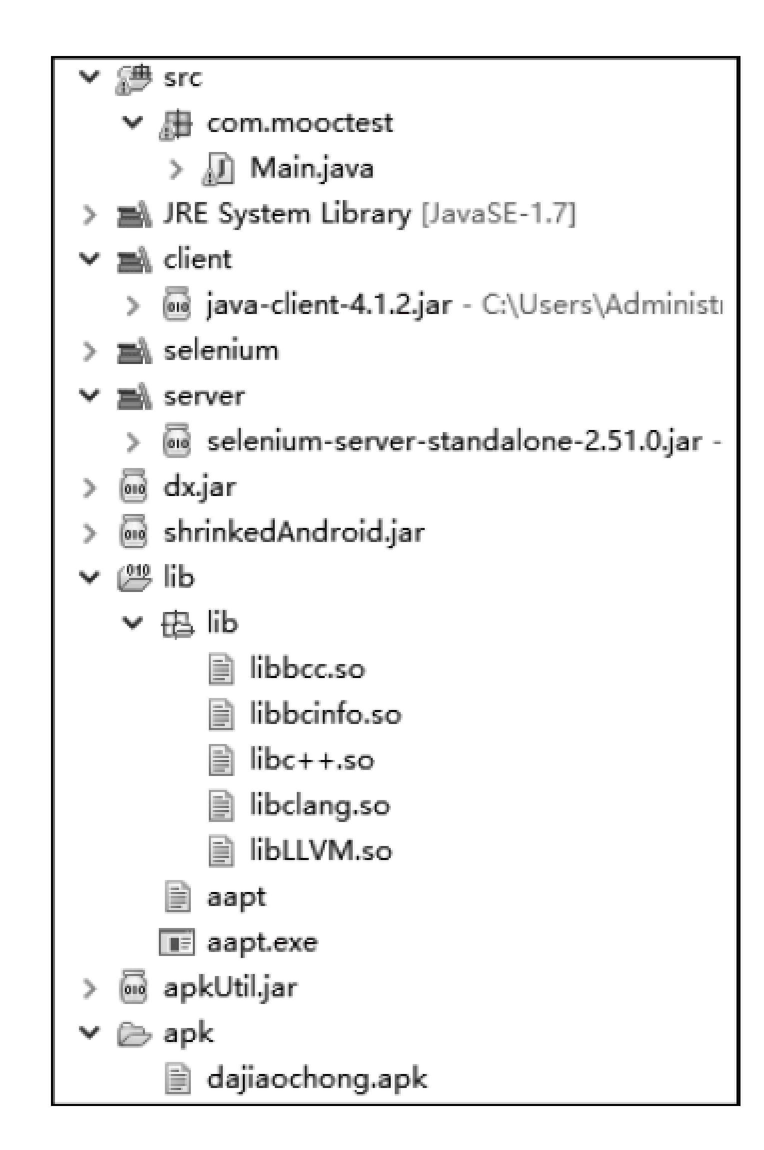


图 9.11　项目结构



图 9.12　打开 USB 调试功能

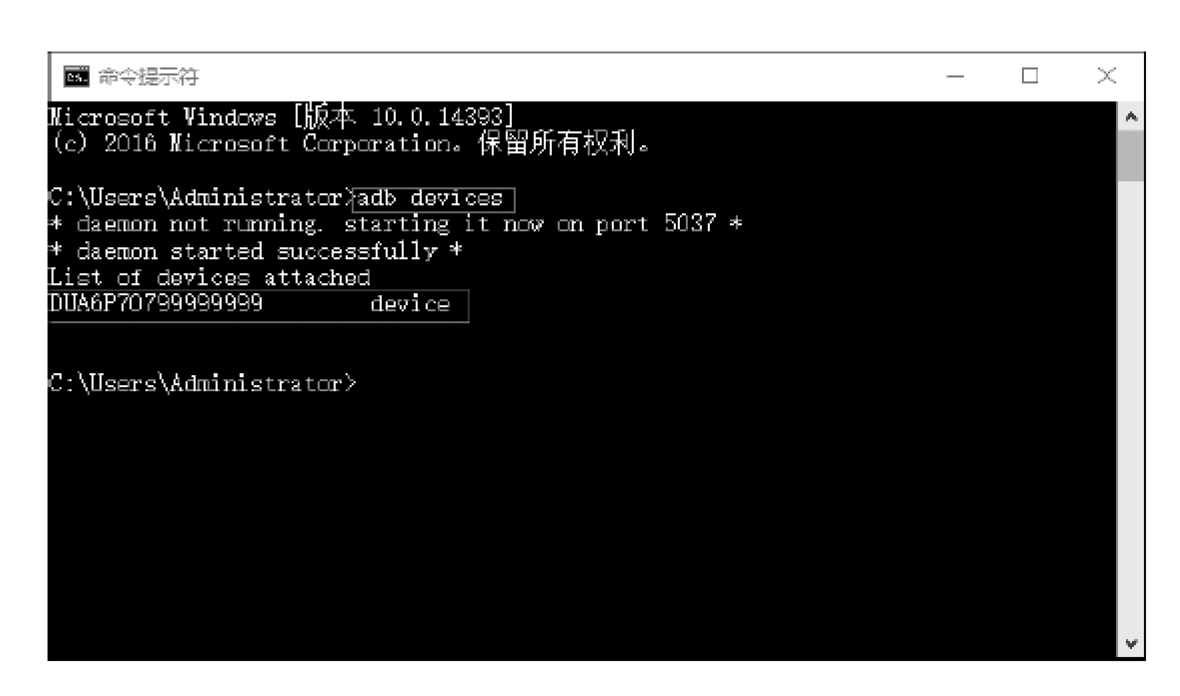


图 9.13　手机 udid 号

步骤 5：打开 Appium，进入设置页面，如图 9.14 所示。

步骤 6：设置 Server Address 为 127.0.0.1，Port 为 8080，如图 9.15 所示。



图 9.14　Appium 的设置页面

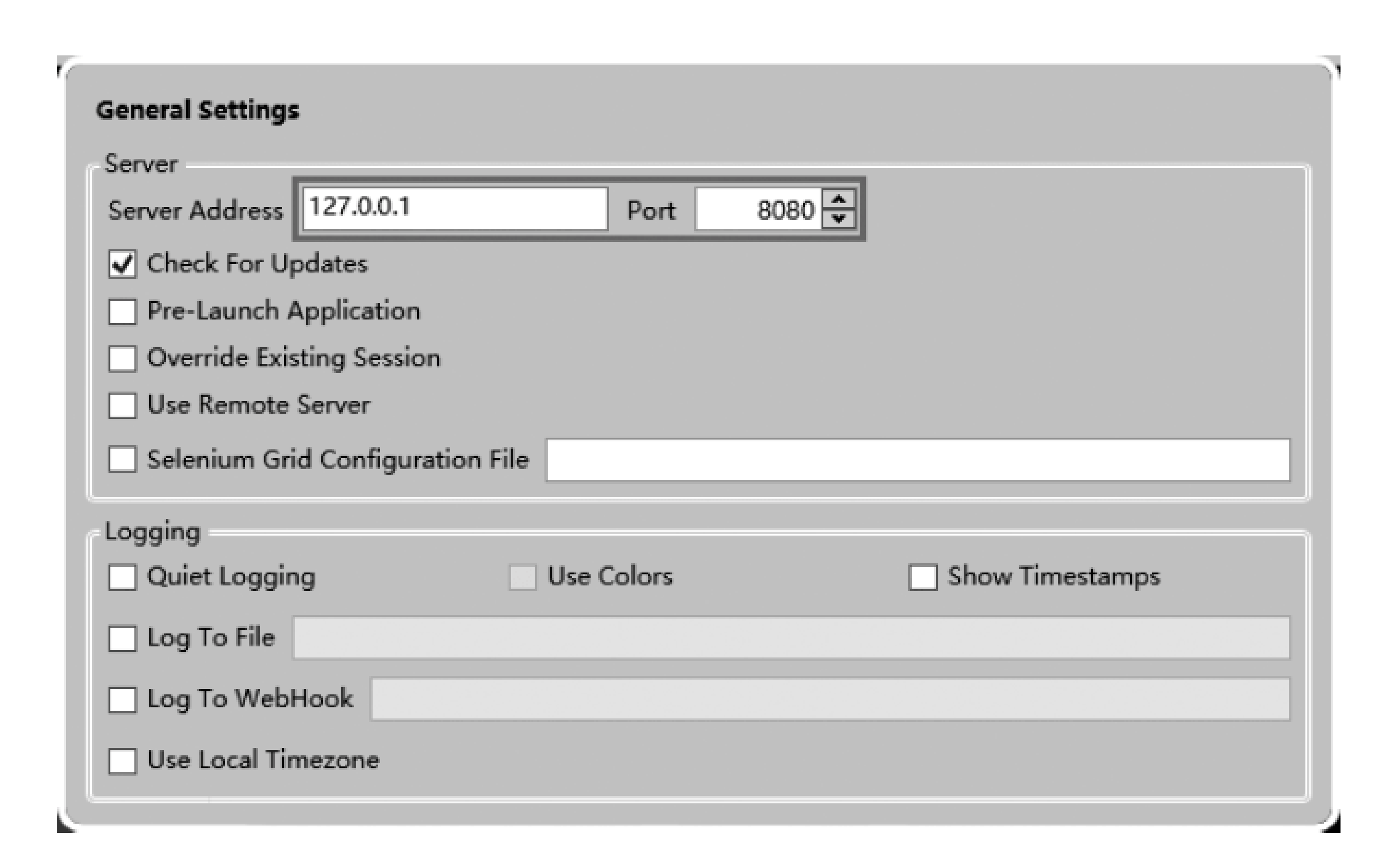


图 9.15　设置服务器网址和端口

步骤 7：启动 Appium，如图 9.16 所示。

步骤 8：编写测试脚本。

```
package com.mooctest。
import com.sinaapp.msdxblog.apkUtil.entity.ApkInfo;
import com.sinaapp.msdxblog.apkUtil.utils.ApkUtil;
import io.appium.java_client.AppiumDriver;
import io.appium.java_client.FindsByAndroidUIAutomator;
import io.appium.java_client.android.AndroidDriver;
import org.omg.CORBA.TIMEOUT;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.remote.CapabilityType;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.UnreachableBrowserException;
import org.openqa.selenium.support.ui.ExpectedCondition;
```



图 9.16　启动 Appium

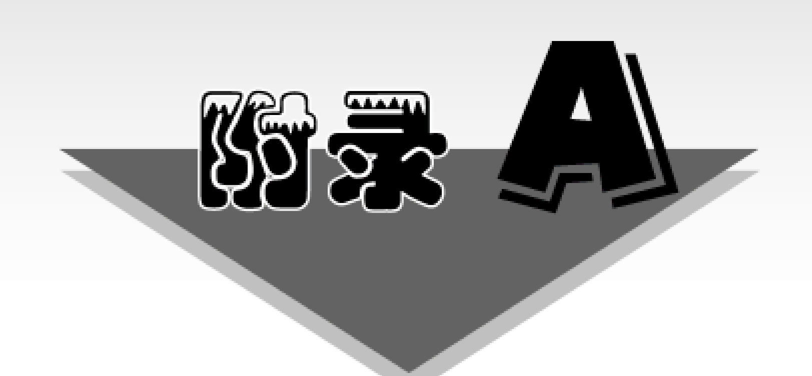
```
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.By;
import java.io.File;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.concurrent.TimeUnit;
import java.util.*;
/**
 * 测试类,只需在 test()函数中完成测试脚本,本次以大角虫软件为例完成登录功能
 * @author Cheng Song
 * @version 1.0.0
 */
public class Main {
    /*
     * port 是在 appium 中配置好的端口,可自行修改
     */
    private String port="8080";
    /*
     * 需要测试的软件 apk 安装包直接替换即可,同时需要将 apk 文件放入 apk 文件夹下
     */
    private String appPath="apk"+File.separator+"dajiaochong.apk";
    /*
     * apk 文件的包名以及启动时的首个 Activity
     */
    private String appPackage;
```

```
private String appActivity;
/*
 * 请将手机的 udid 手动赋值给 deviceUdid
 * 连接好手机后,在 cmd 中通过 adb devices 获得,直接替换
 */
private String deviceUdid="DUA6P70799999999";
//可以直接使用 driver 进行各类操作,所有的测试脚本将在该函数内完成
private void test(AppiumDriver driver){
    System.out.println("正在执行你的脚本逻辑");
    System.out.println("执行脚本");
    /* TODO
     * 以下将使用最常用的几种控件获取方式来演示
     * 1.findElement(By.id("****"))
     * 2.findElement(By.name("****"))
     * 3.findElement(By.className("****"))
     * 4.findElement(By.xpath("****"))
     * 5.切记 UI Automator 中的 index 不能用来定位控件
     * 6.使用 swipe 来完成滑动手势
     */
    /*
     * 1.此处使用 UI Automator 中看到的"我的"按钮的 resource-id 的值
     * "cn.kidstone.cartoon:id/rbtn_mine"来定位控件
     * 如实验步骤中的图示,还可以直接写成 rbtn_mine
     * 然后完成 click()点击事件
     */
      WebElement cancle=driver.findElementById("cn.kidstone.cartoon:id/cancel_
      txt");
      cancle.click();
      WebElement mine=driver.findElement(By.id("cn.kidstone.cartoon:id/rbtn_
      mine"));
      mine.click();
    /*
     * 2.此处使用 UI Automator 中看到的"登录"按钮的 text 的值"登录"来定位控件
     * 然后完成 click()点击事件
     */
      WebElement login=driver.findElement(By.xpath(".//*[@text='登 录']"));
      login.click();
    /*
     * 3.此处使用 UIAutomator 中看到的 class 类来定位控件
     * 由于此处有"用户名"和"密码"两个 EditText 类(第一个是用户名,第二个是密码)
     * 然后使用 sendKeys()依次赋值(此处使用已经注册好的用户名(我的手机号)和密码)
     */
      WebElement username=(WebElement)driver.findElements(By.className("android.
      widget.EditText")).get(0);
```

```
        username.sendKeys("15929949928");
        WebElement password= (WebElement)driver.findElements(By.className("android.
        widget.EditText")).get(1);
        password.sendKeys("123456789");
    /*
     * 4.此处使用 UI Automator 中看到的 xpath 类来定位控件
     * 在 UI Automator 中选中登录按钮,然后从登录按钮依次向其父控件查找
     * 直到找到全局唯一的父控件为止
     * 然后完成 click()点击事件
     */
        WebElement finalLogin = driver. findElement (By. xpath ( "//android. widget.
        ScrollView/" +
        "android.widget.RelativeLayout/" +
        "android.widget.LinearLayout/" +
        "android.widget.Button"));
        finalLogin.click();
    /*
     * 5.此处使用 UI Automator 中看到的 xpath 类来定位“收藏”控件
     * xpath 还能如此使用,By.xpath(".//* [@****]")
     * 然后完成 click()点击事件
     */
    WebElement collect=driver.findElement(By.xpath(".//* [@text='收藏']"));
    collect.click();
    /*
     * 6.获得屏幕的宽和高,然后使用 swipe()来完成滑动,swipe 中的参数含义如下:
     * 起点的 x、y 坐标,终点的 x、y 坐标,滑动的时间(是匀速滑动)
     * swipe(start_point_x,start_point_y,end_point_x,end_point_y,time)
     * 这里向右滑动
     */
    int width=driver.manage().window().getSize().width;
    int height=driver.manage().window().getSize().height;
    driver.swipe(width * 4/5,height/2,width/5,height/2,1000);
}
public static void main(String[] args){
    Main example=new Main();
    example.execute();
}
private void execute(){
    getApkInformation();
    AppiumDriver driver=setUp();
    if(driver !=null){
        test(driver);
    } else {
        System.err.println("服务器未开启");
```

```
        }
    }
    private void getApkInformation(){
        ApkInfo apkInfo=null;
        try {
            apkInfo=new ApkUtil().getApkInfo(appPath);
        } catch(Exception e){
            e.printStackTrace();
        }
        appPackage =apkInfo.getPackageName();
        appActivity=apkInfo.getLaunchableActivity();
        System.out.println("the apk package is "+appPackage+" and the activity is "+
       appActivity);
    }
    private AppiumDriver setUp(){
        File file=new File(appPath);
        String path=file.getAbsolutePath();              //获得 apk 文件的绝对路径
        DesiredCapabilities capabilities=new DesiredCapabilities();
        capabilities.setCapability(CapabilityType.BROWSER_NAME, "");
        /*
         * platformName:设置测试所用的平台类型
         * deviceName:设置设备名称,可以随便取名,最好使用"Android Emulator"
         */
        capabilities.setCapability("platformName", "Android");
        capabilities.setCapability("deviceName", "Android Emulator");
        /*
         * platformVersion:设置测试平台的版本
         * app:设置 apk 文件的绝对路径
         */
        capabilities.setCapability("platformVersion", "4.3");
        capabilities.setCapability("app", path);
        /*
         * appPackage:设置 app 的包名
         * appActivity:设置 app 启动时首个 Activity 名
         * udid:设置连接的手机设备的 id
         */
        capabilities.setCapability("appPackage", appPackage);
        capabilities.setCapability("appActivity", appActivity);
        capabilities.setCapability("udid",deviceUdid);
        /*
         * unicodeKeyboard,resetKeyboard是用来安装 Appium 的输入法的
         * 为了避免手机自带的输入法可能出现的问题,最好设置这两个属性
         */
        capabilities.setCapability("unicodeKeyboard",true);
```

```
            capabilities.setCapability("resetKeyboard",true);
            AppiumDriver driver=null;
            boolean success=false;
            int num=1;
            while(!success && num <=2){
                try {
                    driver=new AndroidDriver<>(new URL("http://127.0.0.1:"+port+"/wd/
                    hub"), capabilities);
                    success=true;
                } catch(MalformedURLException e1){
                    e1.printStackTrace();
                } catch(UnreachableBrowserException e){
                    System.out.println("appium服务器未开启,请手动开启");
                }
                num++;
            }
            /*
             * 隐式时间等待,此处设置将作用于所有控件,用来设置一定的等待时间
             * 防止某些控件还没加载而出现错误
             */
            driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
            return driver;
        }
    }
```



实验报告格式

西安邮电大学

（计算机学院）

课内实验报告

实验名称：________________

专业名称：________________

班级：________________

学生姓名：________________

学号(8 位)：________________

指导教师：________________

实验日期：________________

一、实验目的及实验环境

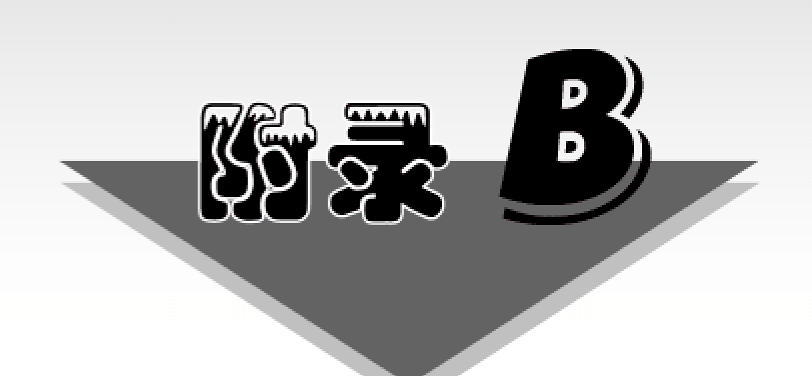
二、实验内容

三、方案设计

四、测试数据及运行结果

五、总结

六、附录：源代码(电子版)



软件测试相关文档模板

软件测试文档是对要执行的软件测试和测试的结果进行描述、定义、规定和报告的任何书面或图示信息。不同类型的测试文档对于软件开发与测试人员的不同角色有用，开发人员希望从软件测试报告的缺陷结果以及分析得到产品开发质量的信息，项目管理者对测试执行中成本、资源和时间更为重视，高层经理希望能够阅读到简单的图表并且能够与其他项目进行同向比较。

与软件测试相关的文档模板一般包括以下内容：

- 测试方案。设计测试什么内容和采用什么样的方法，经过分析，可以得到相应的测试用例列表。
- 测试执行策略。主要包括哪些可以先测试，哪些可以放在一起测试等策略。
- 测试用例。主要根据测试用例列表，写出每一个用例的操作步骤、紧急程度和预期结果。
- 缺陷描述报告。包括测试环境的介绍、预置条件、测试人员、问题重现的操作步骤和测试的现场信息。
- 软件测试报告。目的在于总结测试阶段的工作以及分析测试结果，描述系统是否符合需求。预期读者包括用户、测试人员、开发人员、项目管理者、其他质量管理人员和需要阅读本报告的高层经理。

下面给出软件测试计划模板、软件测试用力设计模板和软件测试报告模板。

B.1 软件测试计划模板

1 引言

1.1 编写目的

本测试计划的具体编写目的，指出预期的读者范围。

1.2 背景

a. 测试计划所从属的软件系统的名称。

b. 说明在开始执行本测试计划之前必须完成的各项工作。

1.3 定义

列出本文件中用到的专门术语的定义。

1.4 参考资料

列出要用到的参考资料,例如:

- 本项目经核准的计划任务书或合同、上级机关的批文。
- 属于本项目的其他已发表的文件。
- 本文件中各处引用的文件、资料,包括要用到的软件开发标准。列出这些文件的标题、文件编号、发表日期和出版单位,说明这些文件资料的来源。

2 计划

2.1 软件说明

说明被测软件的功能、输入和输出等质量指标,作为叙述测试计划的提纲。

2.2 测试内容

列出组装测试和确认测试中的每一项测试内容的名称标识符、这些测试的进度安排以及这些测试的内容和目的,例如模块功能测试、接口正确性测试、运行时间的测试、设计约束和极限的测试等。

2.3 测试 1(标识符)

给出这项测试内容的参与单位及被测试的部位。

2.3.1 进度安排

包括进行测试的日期和工作内容。

2.3.2 条件

陈述本项测试工作对资源的要求,包括:

- 设备。列出类型、数量和预定使用时间。
- 软件。列出用来支持本项测试过程而本身又并不是被测软件的组成部分的软件,如测试驱动程序、测试监控程序、仿真程序、桩模块等。
- 人员。列出在测试工作期间预期可由用户和开发任务组提供的工作人员的人数。

2.3.3 测试资料

列出本项测试所需的资料,例如:

- 有关本项任务的文件。
- 被测试程序及其所在的媒体。
- 测试的输入和输出举例。
- 有关控制此项测试的方法、过程的图表。

2.3.4 测试培训

说明或引用资料说明为被测软件的使用提供培训的计划。规定培训的内容、受训的人员及从事培训的工作人员。

2.4 测试 2(标识符)

……

3 测试设计说明

3.1 测试 1(标识符)

说明对第一项测试内容的测试设计思路。

3.1.1 控制

说明本测试的控制方式,如人工、自动等,以及结果的记录方法。

3.1.2 输入

说明本项测试中所使用的输入数据及选择这些输入数据的策略。

3.1.3 输出

说明预期的输出数据,如测试结果及可能产生的中间结果或运行信息。

3.1.4 过程

说明此项测试的步骤和控制命令,包括测试的准备、初始化、中间步骤和运行结束方式。

3.2 测试 2(标识符)

……

4 评价准则

4.1 范围

说明所选择的测试用例覆盖范围及其局限性。

4.2 数据整理

为了把测试数据加工成便于评价的形式所采用的转换处理技术,如手工方式或自动方式。如果是用自动方式整理数据,还要说明为进行处理而要用到的硬件、软件资源。

4.3 尺度

说明用来判断测试工作是否能通过的评价尺度,如合理的输出结果的类型、测试输出结果与预期输出之间的容许偏离范围、允许中断或停机的最大次数。

4.4 测试人员需求

软件测试计划模板如图 B.1 所示。

类　别	递交时间	建议制定组	建议审批者
单元测试计划	在系统设计阶段就可以起草,最迟可在实现阶段之初递交	开发小组的技术负责人	项目经理
集成测试计划	在系统设计阶段就可以起草,最迟可在实现阶段之初递交	开发小组的技术负责人	项目经理
系统测试计划	在需求开发阶段就可以起草,最迟可在开发工作完成之际递交	独立测试小组的负责人	项目经理
验收测试计划	在需求开发阶段就可以起草,最迟可在系统测试工作完成之际递交	项目经理	客户代表或上级领导

图 B.1 软件测试计划模板

1. 测试范围与目的
2. 测试方法
3. 测试环境与测试辅助工具的描述
4. 测试启动准则与结束准则
5. 应递交的文档

文档名称	预计递交时间
测试计划	
测试用例	
测试报告	

6. 测试人员的组织
7. 任务表

测试任务	开始时间	结束时间	执行人员

8. 可能存在的问题与对象

附录：本计划的审批意见

图 B.1 （续）

B.2 软件测试用例设计模板

1 简介

1.1 编写目的

本测试用例的具体编写目的，指出预期的读者范围。

1.2 定义术语

列出本文件中用到的专门术语的定义。

1.3 参考文献

列出要用到的参考资料，例如：

- 本项目经核准的计划任务书或合同、上级机关的批文。
- 属于本项目的其他已发表的文件。
- 本文件中各处引用的文件、资料，包括要用到的软件开发标准。

2 测试用例设计

2.1 标识符

唯一标识每一个测试用例。

2.2 测试项

准确地描述需要测试的项及其特征。

2.3 测试环境要求

测试用例需要的测试环境。

2.4 输入标准

执行测试用例的输入需求，可能包括数据、文件或者操作。

2.5 输出标准

按照指定的环境和输入标准得到的期望输出结果。

2.6 测试用例之间的关联

标识各测试用例与其他测试用例之间的依赖关系。

在 ANSI/IEEE 829—1983 标准中给出了测试用例编写模板，如表 B.1 所示。

表 B.1 测试用例编写模板

<table>
<tr><td>项目/软件</td><td colspan="2"></td><td>程序版本</td><td colspan="2"></td></tr>
<tr><td>功能模块</td><td colspan="2"></td><td>编制人</td><td colspan="2"></td></tr>
<tr><td>用例编号</td><td colspan="2"></td><td>编制时间</td><td colspan="2"></td></tr>
<tr><td>相关用例</td><td colspan="5"></td></tr>
<tr><td>功能特性</td><td colspan="5"></td></tr>
<tr><td>测试目的</td><td colspan="5"></td></tr>
<tr><td>预置条件</td><td colspan="2"></td><td>特殊规程说明</td><td colspan="2"></td></tr>
<tr><td>参考信息</td><td colspan="5"></td></tr>
<tr><td>测试数据</td><td colspan="5"></td></tr>
<tr><td colspan="6"></td></tr>
<tr><td>操作步骤</td><td>操作描述</td><td>数据</td><td>期望结果</td><td>实际结果</td><td>测试状态(P/F)</td></tr>
<tr><td>1</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>⋮</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>测试人员</td><td></td><td>开发人员</td><td></td><td>负责人</td><td></td></tr>
</table>

B.3　软件测试报告模板

1　简介

1.1　编写目的

本测试报告的具体编写目的，指出预期的读者范围。

1.2　项目背景

对项目目标和目的进行简要说明。

1.3　系统简介

采用框架图对软件设计进行说明。

1.4　术语和缩写词

列出设计本系统/项目的专用术语和缩写语约定。

1.5　参考资料

包括需求分析报告、设计说明书、测试用例报告、用户手册等。

2　测试概要

2.1　测试用例设计

简要介绍测试用例的设计方法，例如等价类划分、边界值分析等方法。

2.2　测试环境与配置

简要介绍测试环境及其配置。

2.3　测试方法和工具

简要介绍测试中采用的方法，讲述测试的重点和测试模式。

3　测试结果及缺陷分析

本部分包括对测试过程的度量和能力评估、对软件产品的质量度量和产品评估。

3.1　测试执行情况与记录

描述测试资源消耗情况，记录实际数据。

3.1.1　测试组织

列出简单的测试组织架构图。

3.1.2　测试时间

列出测试的时间跨度和工作量，最好区分测试文档和活动的时间。数据可供过程度量使用。

3.1.3　测试版本

报告测试次数和回归测试次数，列出子系统/子模块的测试频度。

3.2　覆盖分析

3.2.1　需求覆盖

需求覆盖率是指经过测试的需求/功能和需求规格说明书中所有需求/功能的比值，

通常情况下要达到100%的目标。

3.2.2　测试覆盖

测试覆盖率=执行数/用例总数×100%

3.3　缺陷的统计与分析

测试缺陷说明了实测结果数据和与预期结果数据的偏差,用于表明被测系统的质量。

3.3.1　缺陷汇总

采用缺陷的饼状图和柱状图说明被测系统的严重程度。

3.3.2　缺陷分析

缺陷发现效率=缺陷总数/执行测试用时

用例质量=缺陷总数/测试用例总数×100%

缺陷密度=缺陷总数/功能点总数

缺陷密度可以得出系统各功能或各需求的缺陷分布情况,开发人员可以在此分析基础上得出哪部分功能/需求缺陷最多,从而在今后的开发中注意避免,并注意在实施时予与关注。

测试曲线图描绘被测系统每工作日或周缺陷数情况,得出缺陷走势和趋向。

3.3.3　残留缺陷与未解决问题

描述如何引起缺陷,缺陷的后果,描述造成软件局限性和其他限制性的原因、预防和改进措施、弥补手段和长期策略。

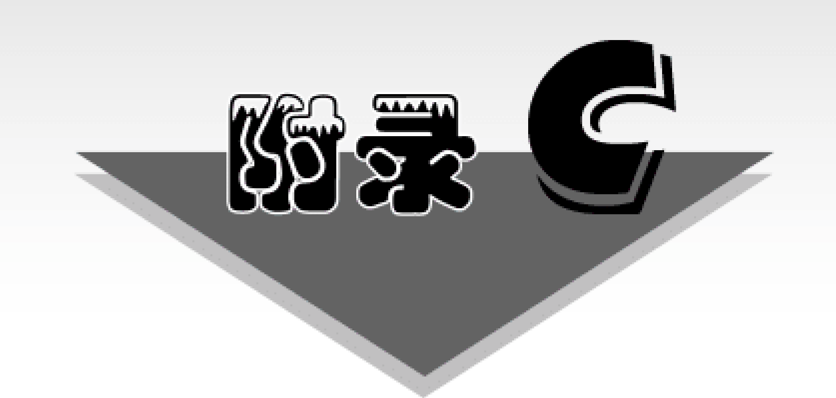
4　测试结论与建议

对上述过程、缺陷分析进行总结。

4.1　测试结论

测试执行是否充分(可以增加对安全性、可靠性、可维护性和功能性的描述),对测试风险的控制措施和成效,测试目标是否完成,测试是否通过,是否可以进入下一阶段项目目标。

4.2　建议



软件测试考试与竞赛简介

本附录主要介绍全国计算机等级考试四级软件测试工程师和全国大学生软件测试大赛。

C.1 全国计算机等级考试四级软件测试工程师

全国计算机等级考试四级软件测试工程师(简称四级软件测试工程师)是全国计算机等级考试中四级的一类,属于计算机技术与软件专业资格(水平)考试的中级。

软件测试工程师考试基本要求如下:

(1) 熟悉软件质量、软件测试及软件质量保证的基础知识。

(2) 掌握代码检查、走查与评审的基本方法和技术。

(3) 掌握白盒测试和黑盒测试的测试用例的设计原则和方法。

(4) 掌握单元测试和集成测试的基本策略和方法。

(5) 了解系统测试、性能测试和可靠性测试的基本概念和方法。

(6) 了解面向对象软件和 Web 应用软件测试的基本概念和方法。

(7) 掌握软件测试过程管理的基本知识和管理方法。

(8) 熟悉软件测试的标准和文档。

(9) 掌握 QESuite 软件测试过程管理平台和 QESat/C++软件分析和工具的使用方法。

C.1.1 考试说明

考试要求如下:

(1) 熟悉计算机基础知识。

(2) 熟悉操作系统、数据库、中间件、程序设计语言基础知识。

(3) 熟悉计算机网络基础知识。

(4) 熟悉软件工程知识,理解软件开发方法及过程。

(5) 熟悉软件质量及软件质量管理基础知识。

(6) 熟悉软件测试标准。

(7) 掌握软件测试技术及方法。

(8) 掌握软件测试项目管理知识。

(9) 掌握C语言及C++或Java语言程序设计技术。

(10) 了解信息化及信息安全基础知识。

(11) 熟悉知识产权相关法律、法规。

(12) 正确阅读并理解相关领域的英文资料。

通过本考试的合格人员能在掌握软件工程与软件测试知识基础上,运用软件测试管理办法、软件测试策略、软件测试技术独立承担软件测试项目,具有工程师的实际工作能力和业务水平。

本考试设置的科目如下:

(1) 软件工程与软件测试基础知识,考试时间为150分钟,笔试,选择题。

(2) 软件测试应用技术,考试时间为150分钟,笔试,问答题。

C.1.2 考试大纲及考试重点

考试科目1:软件工程与软件测试基础知识

1 计算机系统基础知识

1.1 计算机系统构成及硬件基础知识

- 计算机系统的构成
- 处理机
- 基本输入输出设备
- 存储系统

1.2 操作系统基础知识

- 操作系统的中断控制、进程管理、线程管理
- 处理机管理、存储管理、设备管理、文件管理、作业管理
- 网络操作系统和嵌入式操作系统基础知识
- 操作系统的配置

1.3 数据库基础知识

- 数据库基本原理
- 数据库管理系统的功能和特征
- 数据库语言与编程

1.4 中间件基础知识

1.5 计算机网络基础知识

- 网络分类、体系结构与网络协议
- 常用网络设备
- Internet基础知识及其应用
- 网络管理

1.6 程序设计语言知识

- 汇编、编译、解释系统的基础知识
- 程序设计语言的基本成分(数据、运算、控制和传输、过程(函数)调用)

- 面向对象程序设计
- 各类程序设计语言的主要特点和适用情况
- C语言以及C++(或Java)语言程序设计基础知识

2 标准化基础知识

- 标准化的概念(标准化的意义、标准化的发展、标准化机构)
- 标准的层次(国际标准、国家标准、行业标准、企业标准)
- 标准的类别及生命周期

3 信息安全知识

- 信息安全基本概念
- 计算机病毒及防范
- 网络入侵手段及防范
- 加密与解密机制

4 信息化基础知识

- 信息化相关概念
- 与知识产权相关的法律、法规
- 信息网络系统、信息应用系统、信息资源系统基础知识

5 软件工程知识

5.1 软件工程基础
- 软件工程概念
- 需求分析
- 软件系统设计
- 软件组件设计
- 软件编码
- 软件测试
- 软件维护

5.2 软件开发方法及过程
- 结构化开发方法
- 面向对象开发方法
- 瀑布模型
- 快速原型模型
- 螺旋模型

5.3 软件质量管理
- 软件质量及软件质量管理概念
- 软件质量管理体系

- 软件质量管理的目标、内容、方法和技术

5.4 软件过程管理
- 软件过程管理概念
- 软件过程改进
- 软件能力成熟度模型

5.5 软件配置管理
- 软件配置管理的意义
- 软件配置管理的过程、方法和技术

5.6 软件开发风险基础知识
- 风险管理
- 风险防范及应对

5.7 软件工程有关的标准
- 软件工程术语
- 计算机软件开发规范
- 计算机软件产品开发文件编制指南
- 计算机软件需求规范说明编制指南
- 计算机软件测试文件编制规范
- 计算机软件配置管理计划规范
- 计算机软件质量保证计划规范
- 数据流图、程序流程图、系统流程图、程序网络图和系统资源图的文件编制符号及约定

6 软件评测师职业素质要求

- 软件评测师职业特点与岗位职责
- 软件评测师行为准则与职业道德要求
- 软件评测师的能力要求

7 软件评测知识

7.1 软件测试基本概念
- 软件质量与软件测试
- 软件测试定义
- 软件测试目的
- 软件测试原则
- 软件测试对象

7.2 软件测试过程模型
- V 模型
- W 模型
- H 模型

- 测试模型的使用

7.3 软件测试类型

- 单元测试、集成测试、系统测试
- 确认测试、验收测试
- 开发方测试、用户测试、第三方测试
- 动态测试、静态测试
- 白盒测试、黑盒测试、灰盒测试

7.4 软件问题分类

- 软件错误
- 软件缺陷
- 软件故障
- 软件失效

7.5 测试标准

7.5.1 GB/T 16260.1—2003《软件工程 产品质量 第1部分：质量模型》

7.5.2 GB/T 18905.1—2002《软件工程 产品评价 第1部分：概述》

7.5.3 GB/T 18905.5—2002《软件工程 产品评价 第5部分：评价者用过程》

8 软件评测现状与发展

- 国内外现状
- 软件评测发展趋势

9 专业英语

- 正确阅读并理解相关领域的英文资料

考试科目2 软件测试应用技术

1 软件生命周期测试策略

1.1 设计阶段的评审

- 需求评审
- 设计评审
- 测试计划与设计

1.2 开发与运行阶段的测试

- 单元测试
- 集成测试
- 系统(确认)测试
- 验收测试

2 测试用例设计方法

2.1 白盒测试设计

- 白盒测试基本技术
- 白盒测试方法

2.2 黑盒测试用例设计
- 测试用例设计方法
- 测试用例的编写

2.3 面向对象测试用例设计

2.4 测试方法选择的策略
- 黑盒测试方法选择策略
- 白盒测试方法选择策略
- 面向对象软件的测试策略

3 软件测试技术与应用

3.1 软件自动化测试
- 软件自动化测试基本概念
- 选择自动化测试工具
- 功能自动化测试
- 负载压力自动化测试

3.2 面向对象软件的测试
- 面向对象测试模型
- 面向对象分析的测试
- 面向对象设计的测试
- 面向对象编程的测试
- 面向对象的单元测试
- 面向对象的集成测试
- 面向对象的系统测试

3.3 负载压力测试
- 负载压力测试基本概念
- 负载压力测试解决方案
- 负载压力测试指标分析
- 负载压力测试实施

3.4 Web 应用测试
- Web 应用的测试策略
- Web 应用设计测试
- Web 应用开发测试
- Web 应用运行测试

3.5 网络测试
- 网络系统全生命周期测试策略
- 网络仿真技术

- 网络性能测试
- 网络应用测试

3.6 安全测试

- 测试内容
- 测试策略
- 测试方法

3.7 兼容性测试

- 硬件兼容性测试
- 软件兼容性测试
- 数据兼容性测试
- 新旧系统数据迁移测试
- 平台软件测试

3.8 易用性测试

- 功能易用性测试
- 用户界面测试

3.9 文档测试

- 文档测试的范围
- 用户文档的内容
- 用户文档测试的要点
- 用户手册的测试
- 在线帮助的测试

4 测试项目管理

- 测试过程的特性与要求
- 软件测试与配置管理
- 测试的组织与人员
- 测试文档
- 软件测试风险分析
- 软件测试的成本管理

C.1.3 参考资料

1. 张友生. 软件评测师考试考点分析与真题详解. 北京：电子工业出版社，2005.

2. 教育部考试中心. 全国计算机等级考试四级教程 ：软件测试工程师(2008 年版). 北京：高等教育出版社，2007.

3. 王健，苗勇，刘郢. 软件测试员培训教材. 北京：电子工业出版社，2003.

4. 蔡为东. 软件测试工程师面试指导. 北京：科学出版社，2007.

5. 全国计算机等级考试官方网站. http://www.ncre.cn/.

6. 全国计算机等级考试官方论坛. http://bbs.ncre.cn/.

7. 无忧考网计算机等级考试版块. http://www.51test.net/ncre/.
8. 考试吧计算机等级考试版块. http://www.exam8.com/computer/djks/.

C.2　全国大学生软件测试大赛

C.2.1　大赛简介

2016 年,教育部软件工程专业教学指导委员会、中国计算机学会软件工程专业委员会、中国软件测评机构联盟、中国计算机学会系统软件专业委员会和中国计算机学会容错计算专业委员会联合举办首届全国大学生软件测试大赛。大赛旨在建立软件产业和高等教育的资源对接,探索产教研融合的软件测试专业培养体系,进一步推进高等院校软件测试专业建设,深化软件工程实践教学改革。

大赛测试题目来自开源社区和阿里、华为、Testin、途牛等软件企业,全程跟踪和收集选手的软件测试行为。大赛结束后,组委会整理所有大赛数据,脱敏后开放给全球软件工程研究人员,作为教学研究和学术研究的数据集。

全国大学生软件测试大赛主页如图 C.1 所示。



图 C.1　全国大学生软件测试大赛主页

C.2.2　大赛内容

全国大学生软件测试大赛包括如下 3 个部分。

1.“上海软件中心杯”开发者测试大赛

组委会提供来自开源社区的 Java 程序代码,完成 JUnit 测试脚本,同时发布 3 道总决赛题目。

评分规则:

(1) 按分支覆盖率计分。

(2) 按 Bug 的检测比例评分。

2.“慕测杯”移动应用测试大赛

组委会提供来自 Testin 和华为的真实移动应用测试任务，完成测试报告和测试脚本(Appium)。

同时发布两项任务，Testin 任务总分 200 分，华为任务总分 100 分。

Testin 任务评分规则如下：

(1) 按照测试用例(设计文档＋执行日志)与测试需求的匹配度和完整性进行评分(50 分)，系统辅助，以人工判分为准。

(2) 按照测试报告 Bug 描述的准确性和完整性进行评分(50 分)，系统辅助，以人工判分为准。

(3) 按照 Appium 脚本对测试模块对象的覆盖度进行评分(50 分)，自动化判分。

(4) 按照 Appium 脚本的健壮性进行评分，每位选手的测试脚本将在 20 台主流机型上自动执行统计失效率(50 分)，自动化判分。

华为任务评分规则如下：

(1) 按照测试用例(设计文档＋执行日志)与测试需求的匹配度和完整性进行评分(50 分)，系统辅助，以人工判分为准。

(2) 按照测试报告 Bug 描述的准确性和完整性进行评分(50 分)，系统辅助，以人工判分为准。

(3) 不要求写 Appium 测试脚本。

3.“凯云杯”嵌入式测试大赛共 3 个分项赛。

组委会提供来自凯云的嵌入式测试设备及测试任务，完成测试报告和测试脚本(ESITest)。

发布一项任务：嵌入式系统测试。

评分规则如下：

(1) 检错率评分，发现预埋问题(满分 50 分)。

(2) 测试数据质量评价，能够体现选择测试数据的策略(满分 30 分)。

(3) 测试脚本质量评价，能够结合提交的测试数据，通过脚本运行复现测试结果(满分 20 分)。

参 考 文 献

[1] Chris Wysopal，Lucas Nelson，Dino Dai Zovi，等．软件安全测试艺术．程永敬，译．北京：机械工业出版社，2007．

[2] Paul C. Jorgensen. 软件测试．韩柯，杜旭涛，译．北京：机械工业出版社，2007．

[3] Srinivasan Desikan，Gopalaswamy Ramesh．软件测试原理与实践．韩柯，李娜，译．北京：机械工业出版社，2007．

[4] 马瑟．软件测试基础教材(英文版)．北京：机械工业出版社，2008．

[5] Glenford J. Myers，等．软件测试的艺术．王峰，陈杰，译．北京：机械工业出版社，2006．

[6] 段念．软件性能测试过程详解与案例剖析．北京：清华大学出版社，2006．

[7] 董杰．软件测试精要．北京：电子工业出版社，2008．

[8] 贺平．软件测试教程．北京：电子工业出版社，2006．

[9] 宫云战．软件测试教程．北京：机械工业出版社，2008．

[10] 路晓丽，葛玮，龚晓庆，等．软件策划斯技术．北京：机械工业出版社，2007．

[11] 宫云战．软件测试．北京：国防工业出版社，2006．

[12] 朱少民．全程软件测试．北京：电子工业出版社，2007．

[13] 教育部考试中心．全国计算机等级考试四级教程：软件测试工程师(2008 年版)．北京：高等教育出版社，2007．

[14] 秦晓．软件测试．北京：科学出版社，2008．

[15] 佟伟光．软件测试．北京：人民邮电出版社，2008．

[16] 古乐，史九林，等．软件测试案例与实践教程．北京：清华大学出版社，2007．

[17] 蔡为东．软件测试工程师面试指导．北京：科学出版社，2007．

[18] 陈文滨，朱小梅，任冬梅．软件测试技术基础．北京：清华大学出版社，2008．

[19] 刘怀亮，相洪贵．软件质量保证与测试．北京：冶金工业出版社，2007．

[20] 周伟明．软件测试实践．北京：电子工业出版社，2008．

[21] 许育诚．软件测试与质量管理．王慧文，改编．北京：电子工业出版社，2004．

[22] 赵斌．软件测试技术经典教程．北京：科学出版社，2007．

[23] 韩万江．软件工程案例教程．北京：机械工业出版社，2009．

[24] 郁莲．软件测试方法与实践．北京：清华大学出版社，2008．

[25] 全国计算机等级考试新大纲研究组．全国计算机等级考试考纲·考点·考题透析与模拟(2009 版)：四级软件测试工程师． 北京：清华大学出版社，2009．

[26] 希赛 IT 教育研发中心组，韩为，王勇．全国计算机等级考试考纲·考点分析、题解与模拟：四级软件测试工程师．北京：电子工业出版社，2009．

[27] 51Testing 软件测试网．http://www.51testing.com/html/index.html．

[28] 泽众软件．http://www.spasvo.com/．

[29] 考试吧软件评测师版块．http://www.exam8.com/computer/spks/rp/．